Weber
Türschmann

BASIC 3

2. Auflage

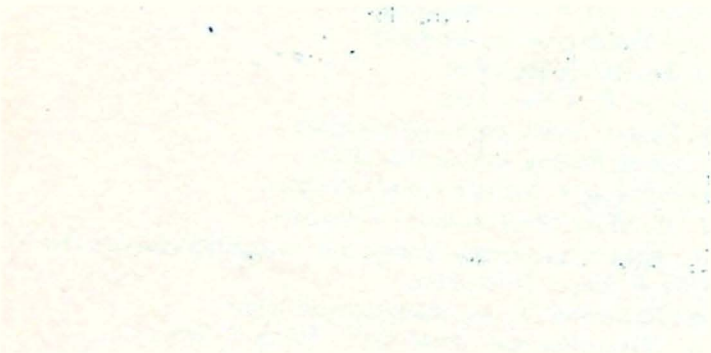
UTB
Haupt

Uni-Taschenbücher 1223

UTB

Eine Arbeitsgemeinschaft der Verlage

Birkhäuser Verlag Basel und Stuttgart
Wilhelm Fink Verlag München
Gustav Fischer Verlag Stuttgart
Francke Verlag München
Harper & Row New York
Paul Haupt Verlag Bern und Stuttgart
Dr. Alfred Hüthig Verlag Heidelberg
Leske Verlag + Budrich GmbH Opladen
J. C. B. Mohr (Paul Siebeck) Tübingen
C. F. Müller Juristischer Verlag – R. v. Decker's Verlag Heidelberg
Quelle & Meyer Heidelberg
Ernst Reinhardt Verlag München und Basel
K. G. Saur München · New York · London · Paris
F. K. Schattauer Verlag Stuttgart · New York
Ferdinand Schöningh Verlag Paderborn · München · Wien · Zürich
Eugen Ulmer Verlag Stuttgart
Vandenhoeck & Ruprecht in Göttingen und Zürich



Karl Weber
Carl Wolfram Türschmann

BASIC 3

Lehr- und Handbuch der Programmiersprache BASIC
mit wirtschaftswissenschaftlichen
Anwendungsbeispielen

2., überarbeitete und erweiterte Auflage

Verlag Paul Haupt Bern und Stuttgart

Die Autoren:

Karl Weber, Dr. oec. publ., M.S. in Accountancy.
1948–1954 Studium der Volks- und Betriebswirtschaftslehre an der Universität Zürich; 1955 Promotion; 1953–1956 Assistent am Handelswissenschaftlichen Seminar der Universität Zürich; 1956–1963 Wissenschaftlicher Mitarbeiter am Betriebswissenschaftlichen Institut der ETH Zürich, mit Unternehmungsberatungstätigkeit im In- und Ausland; 1959 Studium an der London School of Economics and Political Science; 1963–1964 Studium und Tätigkeit als Instructor of Accountancy an der University of Illinois in Urbana; 1964–1967 Assistant Professor of Accountancy an der University of Illinois; 1966 Habilitation an der Universität Zürich; 1966 Fellow in Accountancy an der University of Illinois; 1967–1969 Associate Professor of Accountancy an der University of Illinois; seit 1969 Professor für Betriebswirtschaftslehre an der Justus-Liebig-Universität Giessen.

Carl Wolfram Türschmann, Dipl.-Ing., Dipl.-Oec., Dr. rer. soc.
1964–1970 Studium des Maschinenbaus an der Technischen Hochschule Darmstadt; 1970–1973 Studium der Wirtschaftswissenschaft an der Justus-Liebig-Universität Giessen; 1970–1972 akademischer Tutor am Fachbereich Wirtschaftswissenschaft der Justus-Liebig-Universität; 1972–1977 wissenschaftlicher Mitarbeiter an der Professur für Betriebswirtschaftslehre V der Justus-Liebig-Universität Giessen; 1973–1976 freier Mitarbeiter an der Abteilung für experimentelle Kardiologie am Max-Planck-Institut für physiologische und klinische Forschung (W. G. Kerckhoff-Institut), Bad Nauheim; 1976–1980 Lehrbeauftragter für computergestützte Methoden der empirischen Sozialforschung an der Ruhr-Universität Bochum (Institut für Sportwissenschaft); 1978–1980 Forschungsbeauftragter und Projektleiter des Bundesministeriums für innerdeutsche Beziehungen an der Abteilung für Soziologie der Ruhr-Universität Bochum; 1979 Promotion; seit 1980 Hochschulassistent (Assistenzprofessor) für Betriebswirtschaftslehre an der Justus-Liebig-Universität Giessen.

CIP-Kurztitelaufnahme der Deutschen Bibliothek
Weber, Karl:
BASIC: Lehr- u. Handbuch d. Programmiersprache BASIC
mit wirtschaftswiss. Anwendungsbeispielen /
Karl Weber; Carl Wolfram Türschmann. –
Bern; Stuttgart: Haupt
NE: Türschmann, Carl Wolfram:
3. – 2., überarb. u. erw. Aufl. – 1983.
(UTB für Wissenschaft: Uni-Taschenbücher; 1223)
ISBN 3-258-03232-7
NE: UTB für Wissenschaft / Uni-Taschenbücher

Alle Rechte vorbehalten
Copyright © 1983 by Paul Haupt Berne
Printed in Germany

Vorwort

Das vorliegende Lehr- und Handbuch zur Programmiersprache BASIC stellt das Ergebnis von Lehr- und Forschungstätigkeiten der Professur für Betriebswirtschaftslehre V der Justus-Liebig-Universität Giessen dar.
In Band 1 werden die BASIC-Elementaranweisungen behandelt; Band 2 befasst sich mit den BASIC-Sonderanweisungen. Diese Darstellung wird in Band 3 fortgesetzt. Unter Bezugnahme auf ANSI BASIC werden Graphik- und Diagnoseanweisungen diskutiert und systematisiert. Beispiele zur Anwendung der Sonderanweisungen vermitteln die im vorliegenden Band enthaltenen Applikationsprogramme aus dem Gebiet der Wirtschaftswissenschaft.
Ausser den BASIC-Sprachanweisungen werden in Band 3 – unter spezieller Bezugnahme auf die FOSBIC Version 2 – auch die BASIC-Systemanweisungen behandelt und damit Möglichkeiten zur Entwicklung umfassender BASIC-Programmiersysteme aufgezeigt.
Band 3 enthält auch ein Verzeichnis der wichtigsten Computerherstellfirmen, mit Adressen und detaillierten Angaben über die verfügbaren BASIC-Manuale (Enquete 1982). Ausserdem wird die neuere – insbesondere angloamerikanische – Buchliteratur besprochen.
Die Verfasser stellen sich weiterhin zur Aufgabe, für den deutschen Sprachbereich als Informationsstelle zur BASIC-Standardisierung und als Austauschstelle für BASIC-Programme zu dienen.

Karl Weber
Carl Wolfram Türschmann — Giessen 1983

Die Autoren:

Karl Weber, Dr. oec. publ., M.S. in Accountancy.
1948–1954 Studium der Volks- und Betriebswirtschaftslehre an der Universität Zürich; 1955 Promotion; 1953–1956 Assistent am Handelswissenschaftlichen Seminar der Universität Zürich; 1956–1963 Wissenschaftlicher Mitarbeiter am Betriebswissenschaftlichen Institut der ETH Zürich, mit Unternehmungsberatungstätigkeit im In- und Ausland; 1959 Studium an der London School of Economics and Political Science; 1963–1964 Studium und Tätigkeit als Instructor of Accountancy an der University of Illinois in Urbana; 1964–1967 Assistant Professor of Accountancy an der University of Illinois; 1966 Habilitation an der Universität Zürich; 1966 Fellow in Accountancy an der University of Illinois; 1967–1969 Associate Professor of Accountancy an der University of Illinois; seit 1969 Professor für Betriebswirtschaftslehre an der Justus-Liebig-Universität Giessen.

Carl Wolfram Türschmann, Dipl.-Ing., Dipl.-Oec., Dr. rer. soc.
1964–1970 Studium des Maschinenbaus an der Technischen Hochschule Darmstadt; 1970–1973 Studium der Wirtschaftswissenschaft an der Justus-Liebig-Universität Giessen; 1970–1972 akademischer Tutor am Fachbereich Wirtschaftswissenschaft der Justus-Liebig-Universität; 1972–1977 wissenschaftlicher Mitarbeiter an der Professur für Betriebswirtschaftslehre V der Justus-Liebig-Universität Giessen; 1973–1976 freier Mitarbeiter an der Abteilung für experimentelle Kardiologie am Max-Planck-Institut für physiologische und klinische Forschung (W. G. Kerckhoff-Institut), Bad Nauheim; 1976–1980 Lehrbeauftragter für computergestützte Methoden der empirischen Sozialforschung an der Ruhr-Universität Bochum (Institut für Sportwissenschaft); 1978–1980 Forschungsbeauftragter und Projektleiter des Bundesministeriums für innerdeutsche Beziehungen an der Abteilung für Soziologie der Ruhr-Universität Bochum; 1979 Promotion; seit 1980 Hochschulassistent (Assistenzprofessor) für Betriebswirtschaftslehre an der Justus-Liebig-Universität Giessen.

CIP-Kurztitelaufnahme der Deutschen Bibliothek
Weber, Karl:
BASIC: Lehr- u. Handbuch d. Programmiersprache BASIC
mit wirtschaftswiss. Anwendungsbeispielen /
Karl Weber; Carl Wolfram Türschmann. –
Bern; Stuttgart: Haupt
NE: Türschmann, Carl Wolfram:
3. – 2., überarb. u. erw. Aufl. – 1983.
(UTB für Wissenschaft: Uni-Taschenbücher; 1223)
ISBN 3-258-03232-7
NE: UTB für Wissenschaft / Uni-Taschenbücher

Alle Rechte vorbehalten
Copyright © 1983 by Paul Haupt Berne
Printed in Germany

Vorwort

Das vorliegende Lehr- und Handbuch zur Programmiersprache BASIC stellt das Ergebnis von Lehr- und Forschungstätigkeiten der Professur für Betriebswirtschaftslehre V der Justus-Liebig-Universität Giessen dar.
In Band 1 werden die BASIC-Elementaranweisungen behandelt; Band 2 befasst sich mit den BASIC-Sonderanweisungen. Diese Darstellung wird in Band 3 fortgesetzt. Unter Bezugnahme auf ANSI BASIC werden Graphik- und Diagnoseanweisungen diskutiert und systematisiert. Beispiele zur Anwendung der Sonderanweisungen vermitteln die im vorliegenden Band enthaltenen Applikationsprogramme aus dem Gebiet der Wirtschaftswissenschaft.
Ausser den BASIC-Sprachanweisungen werden in Band 3 – unter spezieller Bezugnahme auf die FOSBIC Version 2 – auch die BASIC-Systemanweisungen behandelt und damit Möglichkeiten zur Entwicklung umfassender BASIC-Programmiersysteme aufgezeigt.
Band 3 enthält auch ein Verzeichnis der wichtigsten Computerherstellfirmen, mit Adressen und detaillierten Angaben über die verfügbaren BASIC-Manuale (Enquete 1982). Ausserdem wird die neuere – insbesondere angloamerikanische – Buchliteratur besprochen.
Die Verfasser stellen sich weiterhin zur Aufgabe, für den deutschen Sprachbereich als Informationsstelle zur BASIC-Standardisierung und als Austauschstelle für BASIC-Programme zu dienen.

Karl Weber
Carl Wolfram Türschmann Giessen 1983

Inhaltsverzeichnis

1 BASIC-Charakteristik 12
11 BASIC-Programmiersysteme 12
2 BASIC-Problembearbeitung 14
21 Problembeschreibung und Problemanalyse 14
22 Programmplanung 16
23 Programmierung 24
24 Programmtest 25
3 BASIC-Sprachstruktur 27
31 Allgemeines 27
32 Elementaranweisungen 28
321 Grundsymbole 28
322 Konstante 28
323 Variable 29
324 Funktionen 31
325 Vereinbarungen 33
325.1 DIM-Statement 33
325.2 Benutzerdefinierte Funktionen 34
325.3 Deklarationen 37
326 Eingabeanweisungen 39
327 Ausgabeanweisungen 41
328 Programmsteuerungsanweisungen 41
328.1 Wiederholungsblöcke 42
328.2 Auswahlblöcke 45
328.3 Ausnahmeblöcke 47
33 Sonderanweisungen 49
331 Matrizenanweisungen 49
332 Fileanweisungen 53
332.1 Organisation und Struktur von Files 54
323.11 Filestruktur 55
332.12 Fileorganisation 56
332.2 Filevereinbarungen 58
332.21 OPEN FL-Statement 58
332.22 CLOSE FL-Statement 60
332.23 ERASE FL-Statement 60
332.3 Fileausgabeanweisungen 60
332.31 Output FL-Statement 61

332.32 Output USING FL-Statement 65
332.33 MAT Output FL-Statement 66
332.34 MAT Output USING FL-Statement . . . 66
332.35 MARGIN FL-Statement 67
332.36 ZONEWIDTH FL-Statement 67
332.4 Filemodifikationsanweisungen 68
332.41 REWRITE FL-Statement 69
332.42 REWRITE WITH FL-Statement 70
332.43 MAT REWRITE FL-Statement 70
332.44 MAT REWRITE WITH FL-Statement . 71
332.45 DELETE FL-Statement 71
332.5 Fileeingabeanweisungen 72
332.51 INPUT FL-Statement 73
332.52 READ WITH FL-Statement 75
332.53 MAT INPUT FL-Statement 76
332.54 MAT READ WITH FL-Statement . . . 77
332.6 Filekontrollanweisungen 78
332.61 RESTORE FL-Statement 78
332.62 INQUIRE FL-Statement 80
333 Erweiterte Programmsteuerungsanweisungen 82
333.1 Subroutineanweisungen 82
333.11 CALL-Statement 83
333.12 SUB-Block 83
333.13 FUNCTION-Block 83
333.2 Programmsegmentierungsanweisungen 85
334 Graphikanweisungen 85
334.1 Koordinatenanweisungen 87
334.11 Koordinatendefinitionsanweisungen . . . 88
334.111 SET BOUNDS-Statement . . . 89
334.112 SET VIEWPORT-Statement . . 91
334.113 SET WINDOW-Statement . . . 92
334.12 Koordinatenabfrageanweisungen 95
334.121 INQUIRE BOUNDS-Statement 95
334.122 INQUIRE VIEWPORT-Statement 96
334.123 INQUIRE WINDOW-Statement 97
334.2 Bildpunktanweisungen 98
334.21 Bildpunktangabeanweisungen 99
334.211 SET BRUSH-Statement 99
334.212 SET CLIP-Statement 100
334.213 CLEAR-Statement 101
334.22 Bildpunktabfrageanweisungen 102

334.211 INQUIRE BRUSH-Statement . 102
334.222 INQUIRE CLIP-Statement . . . 104
334.3 Zeichenanweisungen 104
334.31 Graphische Ausgabeanweisungen 106
334.311 PLOT-Statement 107
334.312 MAT PLOT-Statement 110
334.313 MAT FILL-Statement 113
334.314 SET POSITION-Statement . . . 114
334.315 SET BEAM-Statement 115
334.316 SET JUSTIFY-Statement 115
334.317 SET CHARACTER HEIGHT-Statement 116
334.318 SET TEXT ANGLE-Statement 116
334.319 GRAPHIC PRINT-Statement . 117
334.32 Graphische Abfrageanweisungen 119
334.321 ASK POSITION-Statement . . . 120
334.322 ASK BEAM-Statement 120
334.323 ASK JUSTIFY-Statement . . . 120
334.324 ASK CHARACTER HEIGHT-Statement 121
334.325 ASK TEXT ANGLE-Statement . 121
334.33 Graphische Eingabeanweisungen 122
334.331 GRAPHIC INPUT-Statement . 122
334.4 Bildunterprogrammanweisungen 125
334.41 Call PLOT-Statement 126
334.42 PICTURE-Block 131
335 Diagnoseanweisungen 134
335.1 Systemdiagnoseanweisungen 136
335.11 DEBUG ON-Statement 136
335.12 DEBUG OFF-Statement 137
335.2 Programmdiagnoseanweisungen 137
335.21 TRACE ON-Statement 138
335.22 TRACE OFF-Statement 141
335.23 BREAK-Statement 141
335.24 PRINT ALL-Statement 142
335.25 DUMP-Statement 143
34 Gesamtüberblick 147
4 BASIC-Systemanweisungen 158
41 Systemsteuerungsanweisungen 160
411 BASIC-Initialisierungsanweisungen 162
412 Programmeditierungsanweisungen 163
413 Programmumsetzungsanweisungen 163
414 Programmausführungsanweisungen 165

42 Programmanipulationsanweisungen 165
421 Ausgabeanweisungen 167
422 Korrekturanweisungen 171
423 Löschanweisungen 174
43 Bibliothekverwaltungsanweisungen 176
431 Übergabeanweisungen 178
432 Ladeanweisungen 181
433 Eliminationsanweisungen 182
434 Katalogisierungsanweisungen 183
44 Diagnoseanweisungen 186
45 Cross-Reference-Listen 186
5 BASIC-Anwendungsbeispiele 190
51 Statistik 190
511 Matrizenrechnung 190
512 Input-Output-Analyse 197
513 Multiple lineare Regression 202
52 Betriebswirtschaftslehre 211
521 Input-Output-Analyse 211
522 Kostenabweichungsanalyse 214
523 Lineare Programmierung 222
53 Volkswirtschaftslehre 241
531 Input-Output-Analyse 241
532 Multisektorales Spielmodell 243
533 Modifiziertes Oszallationsmodell nach J. R. Hicks . . . 249
6 BASIC-Literatur 256
61 Handbücher/Manuale 256
611 Angloamerikanische Publikationen 256
612 Kontinentaleuropäische Publikationen 257
62 Lehrbücher 257
621 Angloamerikanische Publikationen 257
622 Kontinentaleuropäische Publikationen 262
63 Normschriften 263
631 Angloamerikanische Publikationen 263
632 Kontinentaleuropäische Publikationen 263
64 Literaturverzeichnis 263
641 Handbücher/Manuale 263
642 Lehrbücher 271
643 Normschriften und übrige Basismaterialien 274
7 Stichwortverzeichnis 277

Einleitung

In Band 1 der vorliegenden Arbeit werden im wesentlichen die BASIC-Elementaranweisungen behandelt. Band 2 befasst sich mit einem Teil der BASIC-Sonderanweisungen, nämlich mit den Matrizen-, File- und erweiterten Programmsteuerungsanweisungen. Diese Darstellung wird in Band 3 fortgesetzt; dabei werden die sich im Anschluss ans ANSI BASIC 82 gegenüber ANSI BASIC 79 ergebenden Unterschiede bezüglich der in Band 1 und 2 behandelten Anweisungen herausgearbeitet.
Die Behandlung der Graphik- und Diagnoseanweisungen erfolgt in Band 3 unter Bezugnahme auf ANSI BASIC 79 und ANSI BASIC 82, so dass sich der Leser insgesamt einen guten Überblick über den gegenwärtigen Stand der BASIC-Standardisierungsbestrebungen verschaffen kann.
Die BASIC-Sonderanweisungen haben sich bisher in der Praxis noch nicht eindeutig im Sinne von ANSI BASIC 79 oder ANSI BASIC 82 durchzusetzen vermocht. BASIC-Anwendern wird deshalb empfohlen, in Ergänzung zur vorliegenden Arbeit auch die einschlägigen BASIC-Manuale der Computerherstellfirmen zu konsultieren und das im Literaturverzeichnis angeführte Adress- und Quellenmaterial entsprechend zu benutzen.
Zur BASIC-Standardisierung ist insbesondere zu verweisen auf:

ANSI, American National Standards Institute. American National Standard for Minimal BASIC. New York 1978.

ECMA, European Computer Manufacturers Association. Standard ECMA-55 Minimal BASIC. Geneva 1978.

ANSI, American National Standards Institute. American National Standard for BASIC. Preliminary Draft. September 1979. [Hanover] 1979.

ANSI, American National Standards Institute. American National Standard for BASIC. Draft Proposed, February 15, 1982. [Hanover] 1982.

Im vorliegenden Text wurde auch mitberücksichtigt:

ANSI, American National Standards Institute. Comments and Responses (Preliminary Draft). June 15, 1982. Hanover 1982.

Das letzterwähnte Arbeitspapier enthält Änderungsvorschläge, die im Rahmen eines Vernehmlassungsverfahrens von Mitgliedern der ANSI-, ECMA- und ISO-BASIC-Kommissionen eingereicht und teilweise zur Integration in ANSI BASIC 82 akzeptiert wurden.

1 BASIC-Charakteristik

11 BASIC-Programmiersystem

Aufgrund der bisherigen Ausführungen – in den Bänden 1 und 2 – kann BASIC als moderne lernorientierte und benutzerfreundliche Programmiersprache charakterisiert werden. Nach Ansicht führender angloamerikanischer Autoren ist BASIC aber nicht nur eine Programmiersprache, sondern ein Programmiersystem.[1] Als solches umfasst BASIC nicht nur die zum Aufbau von BASIC-Programmen dienenden BASIC-Sprachanweisungen, sondern auch die zur Prozessierung und Verwaltung dieser Programme erforderlichen Systemanweisungen.
Die Systemanweisungen können als BASIC-Anweisungen höherer Ordnung aufgefasst werden und sind jedenfalls von den BASIC-Sprachanweisungen scharf zu trennen. Damit ergibt sich für die BASIC-Anweisungen generell folgende Differenzierung:

⟨BASIC-Anweisung⟩ 400 : = ⟨BASIC-Sprachanweisung⟩ 204 | ⟨BASIC-Systemanweisung⟩ 401 | 402

Die Leistungsfähigkeit jedes BASIC-Programmiersystems hängt wesentlich von Anzahl und Struktur der verfügbaren BASIC-Systemanweisungen ab.
Die grundlegenden BASIC-Systemanweisungen werden in Kapitel 4 – unter besonderer Bezugnahme auf die FOSBIC Version 2 – behandelt. Dabei wird weiterhin zwischen Systemsteuerungs-, Programmanipulations- und Bibliothekverwaltungsanweisungen differenziert. Teilweise sind auch einzelne der – im folgenden allerdings einheitlich (Abs. 335) unter den BASIC-Sprachanweisungen erläuterten – Diagnoseanweisungen den BASIC-Systemanweisungen zuzurechnen. Vgl. Abbildung 1.

1 "BASIC is not just a programming language, it is a programming system including a complete, well defined user interface and a filing system." G. M. Bull, W. Freeman and S. J. Garland. Specification for Standard BASIC. Manchester 1973, p. 21. – Das (von der National Computing Center Ltd. herausgegebene und nun vergriffene) Buch wird im folgenden als NCC Standard BASIC zitiert.

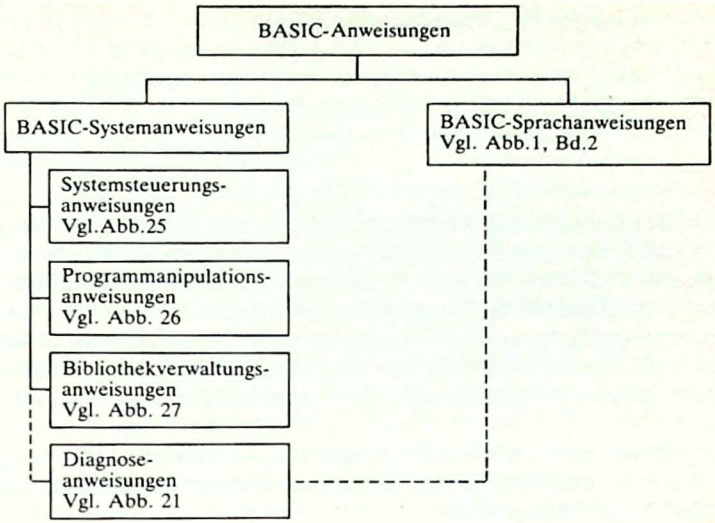


Abbildung 1. BASIC-Programmiersystem

2 BASIC-Problembearbeitung

Die technologische Entwicklung der letzten Jahre hat auf dem Gebiet der EDV das Kostenverhältnis zwischen Hard- und Software von schätzungsweise 85 : 15 im Jahre 1955 auf 25 : 75 zu Beginn der 80er Jahre nahezu umgekehrt. In Zukunft dürfte sich dieses Verhältnis noch weiter in Richtung zunehmender Softwarekosten verändern, so dass um 1985 ein Verhältnis von 15 : 85 erreicht sein wird.[1] Im Zuge dieser Entwicklung kommt dem Einsatz effizienter Software-Entwurfsmethoden immer grössere Bedeutung zu.[2]

Bei Behandlung der BASIC-Problembearbeitung wurde in den Bänden 1 und 2 jeweils nur die traditionelle Programmiermethodik berücksichtigt, die durch die Arbeitsstufen

- Problembeschreibung und -Analyse
- Programmplanung und
- Programmierung

charakterisiert ist. Die folgenden Ausführungen beziehen sich demgegenüber speziell auf neuere Software-Entwicklungsmethoden. Diese werden vor allem auf den ersten beiden Stufen des Problembearbeitungsprozesses eingesetzt.

21 Problembeschreibung und Problemanalyse

Zur Problembeschreibung und -Analyse können die auf den logischen Prozessablauf ausgerichteten BU- (Bottom-Up) und TD- (Top-Down) Methoden sowie deren Varianten benützt werden.[3]

1 Vgl. Barry W. Boehm. Software Engineering: R&D Trends and Defense Needs. In: Peter Wegner (ed.). Research Directions in Software Technology. Cambridge 1979, p. 81.

2 Zur Übersicht vgl. Wolfgang Hesse. Methoden und Werkzeuge zur Software-Entwicklung. Ein Marsch durch die Technologie-Landschaft. Informatik-Spektrum 4 (1981) 4, Oktober, S. 229–245; Erich J. Neuhold. Vergleichende Analyse von Software-Entwurfsmethoden. IHS-Journal 5 (1981) 4, S. 237–259; Lawrence J. Peters and Leonard L. Tripp. Comparing Software Design Methodologies. Datamation 23 (1977) 11, November, p. 89–94.

3 Vgl. Friedemann Singer. Programmieren in der Praxis. Stuttgart 1980, S. 32; Richard C. Linger, Harlan D. Mills and Bernard I. Witt. Structured Programming. Theory and Praxtice. Reading 1979, p. 322–324.

Bei der BU-Methode wird von elementaren Einheiten – insbesondere den hard- und softwaretechnischen Grundvoraussetzungen eines EDV-Systems – ausgegangen und durch Entwicklung von Einheiten höherer Ordnung letztlich die Problemlösung erarbeitet. Die BU-Methode ist zur Bearbeitung maschinennaher, relativ eng abgegrenzter und leicht überschaubarer Probleme gut geeignet.
Bei der TD-Methode werden komplexe Aufgabenstellungen stufenweise in immer kleinere und besser überschaubare Komponenten zerlegt.[4] Im Zuge dieses Prozesses werden von restriktiven Elementarvoraussetzungen unabhängige Lösungen erarbeitet, deren Realisation letztlich allerdings an der Nichtverfügbarkeit hard- und softwaretechnischer Grundvoraussetzungen scheitern kann.
Die agglomerative BU-Methode und die deglomerative TD-Methode führen zu hierarchischen Baumstrukturen, die normalerweise keine Querverbindungen in den einzelnen Ebenen haben. Vgl. Abbildung 2. Auf der obersten Ebene steht die formulierte Aufgabe. Diese ist nach der TD-Methode zunächst in sogenannte Module zu zerlegen. Dabei sollen die Module so konzipiert werden, dass sie unabhängig voneinander programmierbar sind. Sofern für einzelne Module eine Programmierung zunächst noch nicht vorgesehen ist, können sie durch Platzhaltermodule (dummies) ersetzt und damit konzeptionell jedenfalls auf den obersten Ebenen mitberücksichtigt werden. Die Module selbst gliedern sich in Methoden oder Modelle, wobei die Methoden wiederum aus einer Menge von Algorithmen zusammengesetzt sein können. Die Algorithmen teilen sich dann in elementare Funktionen auf. Normalerweise ist eine derartige vierstufige Ebenenaufteilung bei der TD-Analyse ausreichend. Bei komplexen Softwareproblemen kann die Notwendigkeit bestehen, die Strukturierungstiefe der Ebenen und die Anzahl der Ebenen auszuweiten. Hierbei zerlegt man meistens das Gesamtproblem zunächst in Teilprobleme, diese wiederum in kleinere Einheiten. Die Bezeichnung der einzelnen Analysenebenen ist problemorientiert.
Bleibt man bei der vierstufigen Gliederung der Problemanalyse wie sie Abbildung 2 zeigt, so lassen sich die Komponenten der einzelnen Ebenen so mit Programmeinheiten in Verbindung bringen, wie sie eine Programmanalyse ergeben würde. Module würden demnach durch Programmsegmente, Methoden und Modelle durch Subroutinen, Algo-

4 Vgl. Gero Zimmermann. Top-Down-Entwurf eines komplexen Software-Systems. Online – (1981) 9, September, S. 624–625, 628; – (1981) 10, Oktober, S. 732, 734, 736. – Aus der amerikanischen Literatur sei verwiesen auf: Edward Yordon. Managing the Structured Techniques. Englewood Cliffs 1979, p. 58–87; Marvin V. Zelkowitz, Alan C. Shaw and John D. Gannon. Principles of Software Engineering and Design. Englewood Cliffs 1979, p. 50–55.

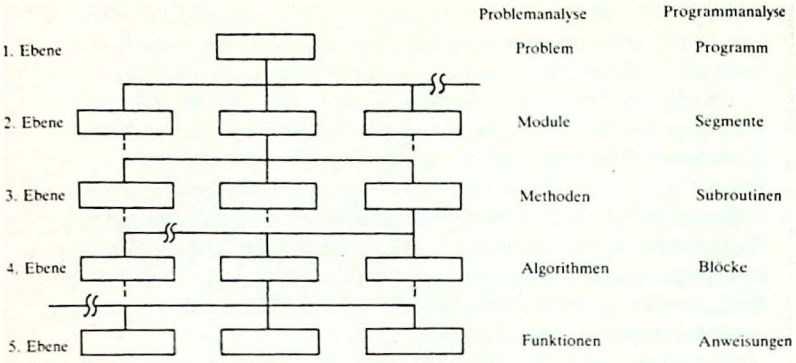


Abbildung 2. Gliederung der Problemanalyse, Top-Down-Methode

rithmen durch Anweisungsblöcke und einzelne Funktionen durch Anweisungen realisiert.
Obwohl BASIC zunächst vollständig ohne Beziehung zu den neueren Softwaretechnologien und insbesondere zur TD-Methode entwickelt wurde, gestattet die vorliegende ANSI BASIC-Norm[5] eine nahezu konforme Programmplanung. Durch die Einführung des CHAIN-Statements[6] können Programme in unabhängige Segmente gegliedert sein. Die Erweiterung der Unterprogrammtechnik mit CALL-Statement und SUB-Block[7] löst das Problem lokaler und globaler Variabler zumindest teilweise. DEF-, IF- und LOOP-Blöcke[8] erlauben die Bildung von Anweisungsfolgen höherer Ordnung (Anweisungsblöcke), die ganze Algorithmen realisieren können.

22 Programmplanung

Auch bezüglich der Programmplanung sind die geläufigen Methoden der Datenfluss- und Programmablaufplanung[9] weiter entwickelt worden. Besonders zu verweisen ist auf die von I. Nassi und B. Shneiderman vorge-

5 ANSI, American National Standards Institute. American National Standard for BASIC. Preliminary Draft. September 1979. Hanover 1979. – Im folgenden zitiert als ANSI BASIC 79.
6 Vgl. Bd. 2, Abs. 332.21.
7 Vgl. Bd. 2, Abs. 333.1.
8 Vgl. Bd. 1, Abs. 331.23, 332.422, 332.44.
9 Vgl. Bd. 1, S. 48–57.

schlagenen Struktogramme.[10] Sie unterstützen zunächst die blockorientierte Betrachtungsweise der TD-Analyse und ergänzen weiterhin die bereits allgemein bekannten und genormten Sinnbilder zur Programmablaufplanung.[11]
Die Struktogramme verzichten vollständig auf die von der Flow-chart-Technik bekannten Verbindungs- bzw. Sprungkonnektoren. Jede Blockstruktur wird als Rechteck dargestellt. Die Blockgrösse kann im Zusammenhang mit der TD-Analyse bis auf Segmentebene angehoben werden. Innerhalb der Programmplanung erscheint es jedoch sinnvoll, die Blockgrösse auf Anweisungsfolgen zu begrenzen, die sich durch strukturierte Anweisungskombinationen realisieren lassen.

Nach dem aus Abbildung 3 ersichtlichen Struktogramm wird eine Subroutine als eine Sequenz von Elementarblöcken dargestellt.

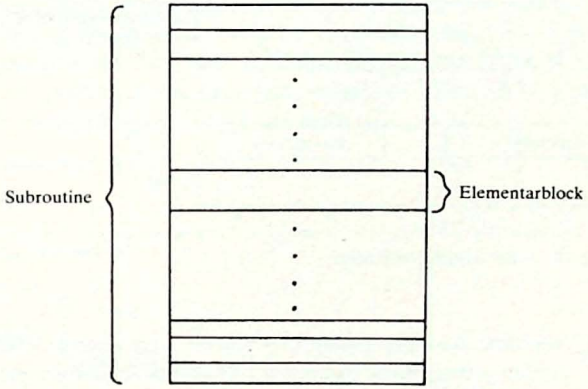


Abbildung 3. Subroutine als Sequenz von Elementarblöcken

10 I. Nassi and B. Shneiderman. Flowchart Techniques for Structured Programming. Sigplan Notices 8 (1973) 1, p. 12–19. – Eine Übersicht über vorgeschlagene graphische Darstellungsformen der Software findet sich bei L. J. Peters and L. L. Tripp. Software Design Representation Schemes. In: Proceedings of the Symposium on Software Engineering. New York 1976, p. 31–56.
11 Vgl. Bd. 1, S. 51.

Bei den Elementarblöcken lassen sich die drei Formen
- Anweisungsblock
- Auswahl- oder Bedingungsblock und
- Wiederholungsblock

unterscheiden.[12]
Der Anweisungsblock ist normalerweise eine lineare Folge von Statements. Sein Symbol ist ein Rechteck; graphische Zusätze können Kommentare enthalten oder einen Anweisungsblock als Programmeinheit höherer Ordnung charakterisieren. Vgl. Abbildung 4.

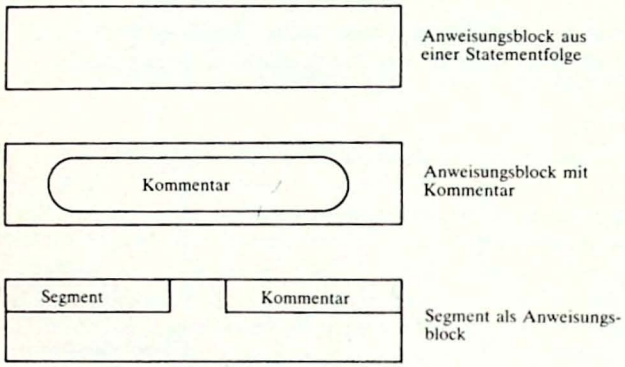


Abbildung 4. Anweisungsblöcke

In BASIC werden Anweisungsblöcke durch sogenannte DEF-Blöcke realisiert,[13] wobei jeder Block mit einem DEF-Statement beginnt und einem DEFEND-Statement endet.
Für Bedingungsblöcke ist charakteristisch, dass sie aus zwei parallel nebeneinander liegenden Abschnitten bestehen, die programmtechnisch jeweils in sequentieller Folge zu realisieren sind.
In BASIC wird diese bedingte Blockstruktur durch den IF-Block, bestehend aus einer Kombination von IF-, ELSE- und ENDIF-Statement, verwirklicht.[14] Vgl. Abbildung 5.

12 Vgl. Peter Schnupp und Christiane Floyd. Software. Programmentwicklung und Projektorganisation. Berlin 1976, S. 47–63.
13 Vgl. Bd. 1, Abs. 331.23, insbesondere S. 151, 153, 157.
14 Vgl. Bd. 1, Abs. 332.422.

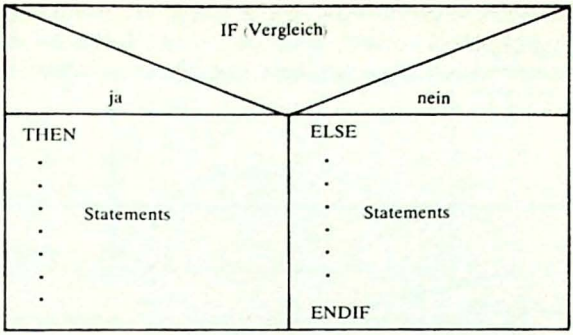


Abbildung 5. Bedingungsblock

Der Auswahlblock unterteilt einen Strukturblock in eine durch die Auswahlbedingung festgelegte Anzahl von Blockteilen. Er entspricht damit einem bedingten GOTO- oder GOSUB-Statement.[15] Von ANSI BASIC wird hierzu speziell die Statementkombination SELECT, CASE, END SELECT vorgeschlagen.[16] Vgl. Abbildung 6.

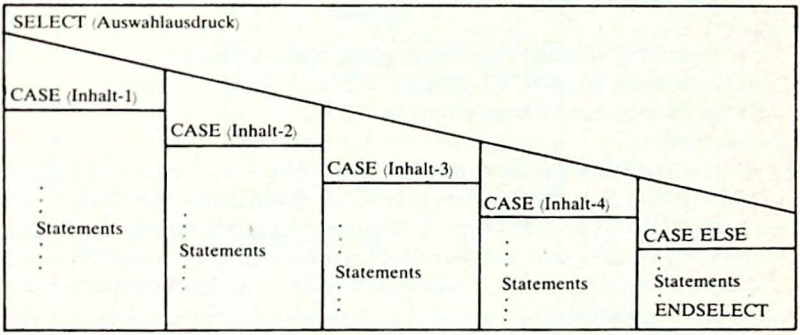


Abbildung 6. Auswahlblock

15 Vgl. Bd. 1, Abs. 332.412 und Abs. 332.472.
16 ANSI BASIC 79, p. 69–71.

Der Auswahlblock kann auch durch eine Folge von ineinandergeschachtelten Bedingungsblöcken und damit in BASIC durch IF-, ELSE- und ENDIF-Statementkombinationsfolgen aufgebaut werden; vgl. Abbildung 7.

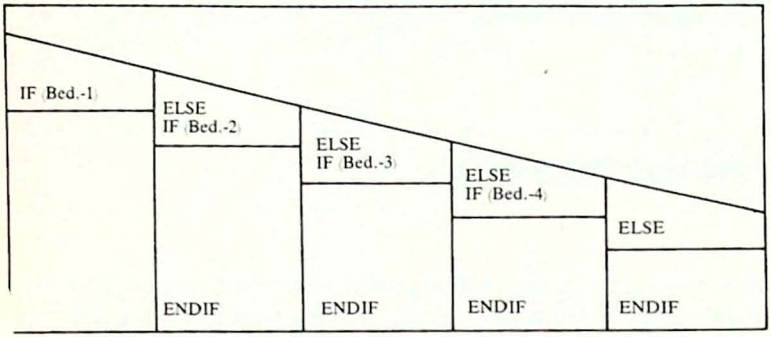


Abbildung 7. Auswahlblock, Alternativform

Bei den Wiederholungsblöcken lassen sich die drei grundsätzlichen Formen
- Wiederholung solange eine Bedingung nicht erfüllt ist
- Wiederholung bis eine Bedingung erfüllt ist und
- Wiederholung mit Abbruchbedingung

unterscheiden.
In einigen BASIC-Dialekten werden diese Wiederholungsblöcke durch spezielle WHILE- oder REPEAT WHILE-Statements, kombiniert mit einem ENDWHILE- oder UNTIL-Statement als Blockende, realisiert. Nach ANSI BASIC sind alle drei Blockarten durch entsprechende Kombinationen von LOOP-, IF-, EXIT- und ENDLOOP-Statements darstellbar.[17] Vgl. Abbildung 8.

17 Zum LOOP-, EXIT- und ENDLOOP-Statement vgl. Abs. 328.1 und Bd. 1, Abs. 332.44.

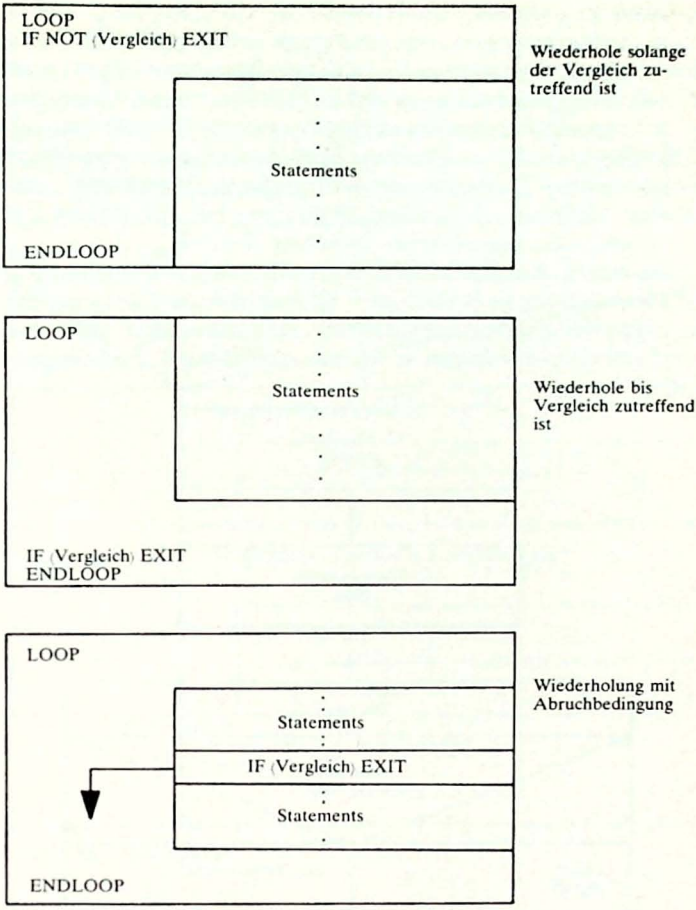


Abbildung 8. Wiederholungsblöcke

Die Entwicklung der Struktogrammsymbole ist noch nicht abgeschlossen; ebenso fehlt eine Normierung. Dem Leser wird empfohlen, sich an die aufgezeigten Darstellungsweisen zu halten.
Im Zusammenhang mit der strukturierten Programmierung und der Entwicklung von Struktogrammen ist eine lebhafte Diskussion um die programmtechnische Realisation derartig geplanter Programme entstanden. Beachtenswert ist insbesondere der Beitrag von Donald E. Knutz, nach dessen Auffassung Programme mit GOTO-Statements – im Gegensatz

zur pointiert geäusserten Meinung von Edsger W. Dijkstra[18] – keineswegs generell als unstrukturiert und veraltet zu gelten haben.[19] Es erscheint deshalb auch nicht unbedingt sinnvoll oder gar notwendig, von der herkömmlichen Programmierweise mit GOTO-Statements abzurücken und auf neue Progammiersprachen umzusteigen.[20] Im Gegenteil muss festgestellt werden, dass die strukturierte Programmplanung – in Verbindung mit den tradierten Erkenntnissen zur Programmentwicklung – eine hervorragende Methode zur übersichtlichen und gut abgestuften Bearbeitung grösserer und komplexerer Probleme darstellt.
Die vorstehende Aussage trifft auch im Hinblick auf eine vorgesehene Programmcodierung in BASIC zu.[21] So lässt sich die – auch mit BASIC-Elementaranweisungen gut codierbare – Matrixinversion[22] zunächst in einem einfachen Struktogramm in der aus Abbildung 9 ersichtlichen Form darstellen.

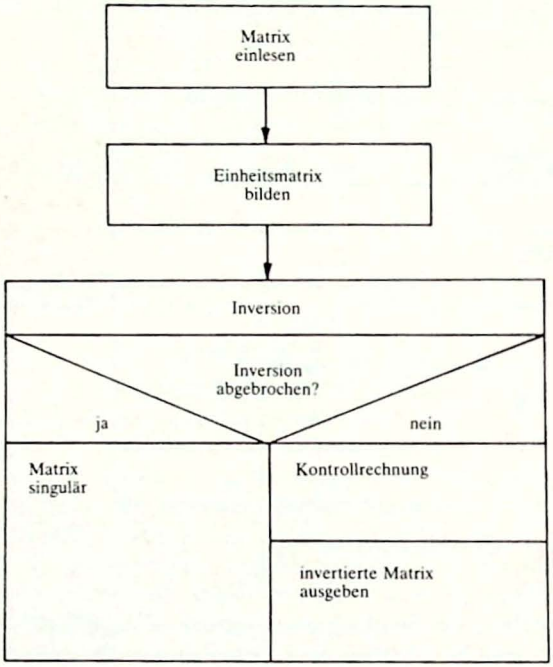


Abbildung 9. Struktogramm der Matrixinversion

18 Edsger W. Dijkstra. The Humble Programmer. Communications of the ACM 15 (1972) 10, October, p. 859-866.
19 Donald E. Knuth. Structured Programming with GOTO Statements. Computing Surveys 6 (1974) 4, December, p. 261–301.

Für die Inversion ergibt sich im Detail der in Abbildung 10 dargestellte Strukturblock. Aus diesem Struktogramm ist ersichtlich, dass der Programmabschnitt zur Kontrolle der Inversion oder zur Mitteilung über eine singuläre Matrix innerhalb des Inversionsblocks erscheint. Diese Schwierigkeit tritt bei Wiederholungsschleifen mit Abbruchbedingung immer auf, wenn vermieden werden soll, dass nach dem Wiederholungsblock die Abbruchbedingung noch einmal abgefragt werden soll.[23]

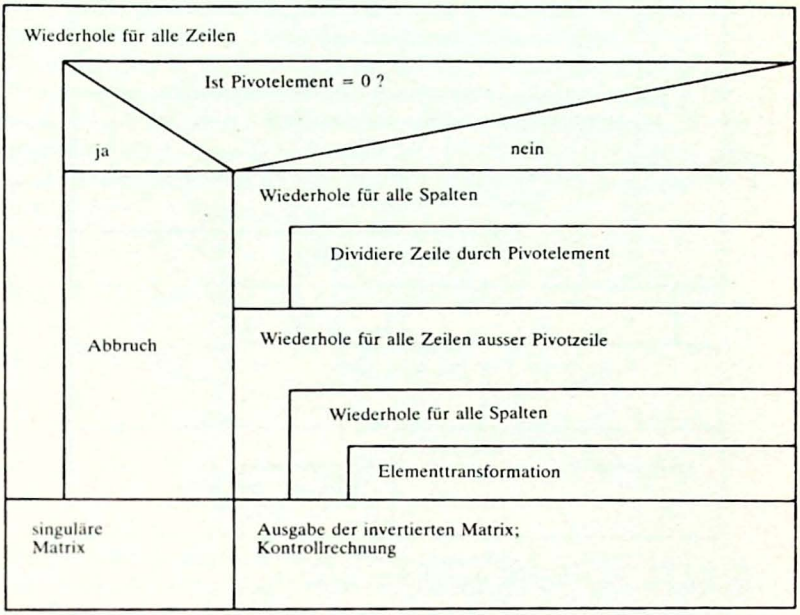


Abbildung 10. Inversion

20 Zu dieser Diskussion vgl. auch: W. A. Wulf. A Case Against the GOTO. Proceedings of the ACM Annual Conference Boston, Mass., August 1972, p. 791–797; Martin E. Hopkins. A Case for the GOTO. Proceedings of the ACM Annual Conference Boston, Mass., August 1972, p. 787–790; Brian W. Kernighan and P. J. Plauger. Programming Style. Examples and Counterexamples. Computing Surveys 6 (1974) 4, December, p. 303–319.

21 Vgl. auch: David H. Ahl. Pascal, Ada and Computer Literacy. An Eclective Interview with Art Luehrmann. Creative Computing 7 (1981) 11, November, p. 116, 118; Al Evans. Side by Side: Pascal and Basic. Creative Computing 7 (1981) 11, November, p. 158, 160–163.

22 Vgl. Bd. 2, Abs. 23.

23 Vgl. Gustav Kastner. Eine Verbesserung der Struktogramm-Symbole. Angewandte Informatik 23 (1981) 2, Februar, S. 68–70.

Die Inversion in getrennten Blöcken ist aus Abbildung 11 ersichtlich.

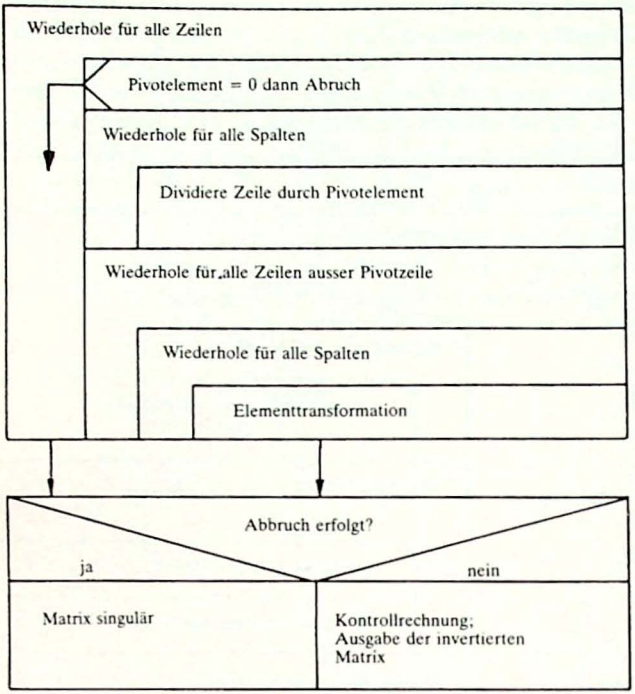


Abbildung 11. Inversion in getrennten Blöcken

Nach diesem Struktogramm wird die Abbruchbedingung im Anschluss an die eigentliche Inversion noch einmal abgefragt. Die damit verbundene Unübersichtlichkeit entfällt bei Verwendung von Abbildung 10.

23 Programmierung

In der Phase der Programmierung entsteht die Aufgabe, die im Rahmen der Programmplanung entwickelten – in einem übersichtlichen Graph zusammenhängend dargestellten – Struktogrammblöcke in die Programmanweisungen der ausgewählten Programmiersprache, hier BASIC, umzusetzen. Diese Aufgabe soll wiederum durch eine systematische, nach Möglichkeit ebenfalls strukturierte, Vorgehensweise gelöst werden.

Da die zu bearbeitenden Komponenten eines Struktogramms kleinere Einheiten – Blöcke – betreffen, können sie nach der tradierten Flowchart-Technik behandelt werden, ohne dass es dabei zu unübersichtlichen Programmabläufen kommen muss. Auch die Verwendung von GOTO-Statements innerhalb von Blöcken ist nunmehr sinnvoll, da hier Ausgangspunkt eines Sprungbefehls und Zielanweisung eng beieinander liegen.
Bei der Programmierung wird mithin für jeden in der Struktogrammanalyse (Makroanalyse) angegebenen Block eine spezielle TD-Analyse bis zur einzelnen BASIC-Anweisung (Mikroanalyse) durchgeführt.
Die Programmierung des Struktogrammblocks der Inversion nach Abbildung 9 und 10 liefert folgenden BASIC-Programmabschnitt, wobei jetzt gegenüber der – aus Bd. 2, Abs. 23, ersichtlichen – Ursprungsversion eine weitere FOR/NEXT-Schleife mit einem IF-Block eingeführt wird.

```
 80 FOR M1=1 TO N1
 90     IF D(M1,M1)=0 GOTO 360
100     T=D(M1,M1)
110     FOR N=1 TO N1
120         D(M1,N)=D(M1,N)/T
130         E(M1,N)=E(M1,N)/T
140     NEXT N
150     FOR M=1 TO N1
160         IF M=M1 THEN
165           REM HIER IST EIN LEERBLOCK
170         ELSE
180           T=D(M,M1)
190           FOR N=1 TO N1
200               D(M,N)=D(M,N)-D(M1,N)*T
210               E(M,N)=E(M,N)-E(M1,N)*T
220           NEXT N
230         ENDIF
240     NEXT M
250 NEXT M1
```

Für die Übersichtlichkeit eines Programms ist es von besonderer Bedeutung, dass die einzelnen Blöcke des Struktogramms in der Anweisungsfolge weiterhin erkennbar bleiben. Dies wird erzielt, indem alle Wiederholungen sowie Bedingungsblöcke mit ihren untergeordneten Anweisungsfolgen eingerückt werden. Obwohl dies von vielen Programmierern zunächst nur als Schönschreiben eines Programms angesehen wird, ist diese Vorgehensweise heute bei umfangreichen BASIC-Programmen unumstritten.

24 Programmtest

Beim Programmtest bietet die Vorgehensweise nach der TD- oder BU-Methode wiederum alternative Möglichkeiten mit unterschiedlichen Vor- und Nachteilen.
Bei der TD-Methode muss das gesamte Problem in Programmstruktur vorliegen, auch wenn es möglich ist, Teilkomponenten oder einzelne

Strukturblöcke durch sogenannte Platzhalter (dummies) zu ersetzen, die erst später ergänzt werden. Jede dieser Ergänzungen macht weitere Programmtests notwendig und kann zu Integrationsfehlern führen.
Bei der BU-Methode werden kleine unabhängige Programmteile getestet und der weitere Programmaufbau erst nach deren Korrektheit fortgesetzt. Die Integration neuer Komponenten kann aber hierbei ebenfalls zu Fehlern führen, sobald nicht mehr gewährleistet ist, dass die integrierten oder ergänzten Programmkomponenten von den übrigen unabhängig sind.
Die Vor- und Nachteile der beiden Methoden führen dazu, sie gleichzeitig einzusetzen, wobei umfangreiche und komplexe Probleme durch die TD-Methode in der Programmplanung und Programmierung in kleine überschaubare Module aufgespaltet werden, die unabhängig vom Gesamtproblem programmiert und getestet werden. Die zahlreichen Module eines Grossproblems können gleichzeitig bearbeitet werden, was zu Zeitersparnissen führen kann. Beim Gesamttest ist nur noch das Integrationsproblem zu lösen.
BASIC unterstützt den Programmtest durch das CHAIN- und LINK-Statement.[24] Beim CHAIN-Statement können entsprechend der TD-Analyse und den dabei gefundenen hierarchischen Baumstrukturen analoge Aufrufstrukturen von BASIC-Programmen erstellt werden, die aus jeweils selbständigen BASIC-Programmen bestehen und einzelne abgegrenzte Problemlösungen liefern.
Das LINK-Statement ist bei BU-Programmtests sinnvoll einsetzbar, indem mit ihm vorab getestete BASIC-Programmteile aus einer Programmbibliothek zu grösseren Programmen temporär oder auch permanent zusammengesetzt werden können.

24 Vgl. Bd. 2, Abs. 332.21.

3 BASIC-Sprachstruktur

31 Allgemeines

Die softwaretechnische Entwicklung der letzten Jahre hat auch bezüglich der BASIC-Sprachstruktur zu sich rasch folgenden Neuvorschlägen zur Standardisierung der Elementar- und Sonderanweisungen geführt. Verwiesen sei insbesondere auf ANSI BASIC 79[1] und ANSI BASIC 82[2], die nicht nur wesentlich über ANSI Minimal BASIC[3] hinausgehen, sondern teilweise auch gegenseitig grundlegend voneinander abweichen.
Bezüglich der Elementaranweisung bringt ANSI BASIC 82 nur relativ geringe Änderungen; die Erweiterungen gegenüber ANSI Minimal BASIC basieren fast ausnahmslos auf den bisherigen Anweisungsformen oder lassen sich durch diese realisieren. ANSI Minimal BASIC stellt damit eine Untermenge von ANSI BASIC 82 dar und beinhaltet weiterhin den BASIC-Nukleus.[4]
Zwischen ANSI BASIC 79 und ANSI BASIC 82 lassen sich vor allem hinsichtlich der Sonderanweisungen markante Unterschiede feststellen. Insbesondere erfährt die Gruppe der Fileanweisungen umfangreiche Erweiterungen und Umgestaltungen in der Kennwortstruktur.[5]
In den folgenden Abschnitten werden die nach ANSI BASIC 82 vorgesehenen wichtigsten Neuerungen bezüglich der einzelnen Anweisungsgruppen vorgetragen und, soweit wie möglich, auch die entsprechenden Strukturgleichungen in geänderter Form aufgeführt. Falls jedoch nach ANSI BASIC 82 im Vergleich zu ANSI BASIC 79 völlig neue Statement-

1 ANSI, American National Standards Institute. American National Standard for BASIC. Preliminary Draft. September 1979. Hanover 1979. – Im folgenden zitiert als ANSI BASIC 79.
2 ANSI, American National Standards Institute. American National Standard for BASIC. Draft Proposed. February 15, 1982. Hanover 1982. – Im folgenden zitiert als ANSI BASIC 82.
3 ANSI, American National Standard Institute. American National Standard for Minimal BASIC. New York 1978. – Im folgenden zitiert als ANSI Minimal BASIC.
4 Vgl. Bd. 1, S. 65–245.
5 Vgl. Bd. 2, Abs. 332. – Die Darstellung der Fileanweisungen in Bd. 2 basiert vollumfänglich auf ANSI BASIC 79.

formen auftreten, werden diese vorerst noch nicht in das in dieser Arbeit verwendete Gleichungssystem integriert. Dies deshalb, weil eine vollständige Verdrängung von ANSI BASIC 79 durch ANSI BASIC 82 fraglich erscheint und eher mit einer integrativen Langzeitwirkung der beiden Normvorschläge zu rechnen ist.

32 Elementaranweisungen

Die BASIC-Elementaranweisungen werden durch ANSI/ECMA Minimal BASIC normiert. Einzelheiten sind aus Band 1 ersichtlich; vgl. Abbildung 19, Bd. 1.
Die folgenden Abschnitte beziehen sich auf ANSI BASIC 82.

321 Grundsymbole

ANSI BASIC 82 trennt die Buchstaben nicht mehr grundsätzlich in Gross- und Kleinbuchstaben[6], sondern spricht einheitlich von Buchstaben im Sinne der ASCII-Zeichen 4/1 bis 5/14 und 6/1 bis 7/10.[7] Die in Band 1 aufgeführte Gleichung 3 entspricht implizit ebenfalls dieser Differenzierung und kann beibehalten werden.

322 Konstante

Durch die generelle Gleichschaltung der Gross- und Kleinbuchstaben kann nun bei den E-formatierten Ganz- und Gleitkommazahlen im Exponententeil anstelle des sonst üblichen Grossbuchstabens E auch der Kleinbuchstabe (e) verwendet werden. Die entsprechenden Strukturgleichungen lauten[8] nunmehr:

$$\underset{25}{\langle \text{E-formatierte Ganzzahl} \rangle} ::= \underset{22}{\langle \text{I-formatierte Ganzzahl} \rangle} \; \underset{29}{\langle \text{Exponententeil} \rangle}$$

6 Vgl. Bd. 1, Abs. 321.1.
7 ANSI BASIC 82, p. 11.
8 Vgl. Bd. 1, Abs. 322 und ANSI BASIC 82, p. 20.

⟨E-formatierte Gleitkommazahl⟩ 30 : = ⟨F-formatierte Gleitkommazahl⟩ 26 ⟨Exponententeil⟩ 27 29

wobei die Änderung nur den Exponententeil betrifft:

⟨Exponententeil⟩ 29 : = E⟨I-formatierte Ganzzahl⟩ 25 | 30 22 | e⟨I-formatierte Ganzzahl⟩ 22

323 Variable

Bei den Variablen läßt ANSI BASIC 82 in der Namensfindung eine erhebliche Erweiterung zu, als Variablennamen nunmehr aus bis zu 31 Buchstaben und Ziffern einschliesslich eines Sonderzeichens gebildet werden können.[9] Das erste Zeichen muss ein Buchstabe sein.[10] Als Sonderzeichen wird nur der untergesetzte Strich ("underline"-Zeichen, im ASCII-Code 5/15) zugelassen.[11] Als Strukturgleichung für die Variable gilt daher gemäss ANSI BASIC 82:[12]

⟨Einfache numerische Variable⟩ 38 36 : = ⟨Name⟩ 41

⟨Indizierte numerische Variable⟩ 39 36 : = ⟨Name⟩ 41 ⟨Indexliste⟩ 42

⟨Einfache alphanumerische Variable⟩ 44 35 : = ⟨Name⟩ 41 $ | ⟨Name⟩ 41 $ ⟨Teilzeichenkettenbegrenzung⟩ 47

9 ANSI BASIC 82, p. 18. "Identifers shall contain at most 31 characters."
10 ANSI BASIC 82, p. 18.
11 ANSI BASIC 82, p. 18 und p. 205.
12 Vgl. Bd. 1, S. 84–91 und ANSI BASIC 82, p. 18, p. 22 und p. 37.

⟨Indizierte alphanumerische Variable⟩ (46, 44) : = ⟨Name⟩ (41) $ ⟨Indexliste⟩ (42) |

⟨Name⟩ (41) $ ⟨Indexliste⟩ (42) ⟨Teilzeichenkettenbegrenzung⟩ (47) |

⟨Name⟩ (41) $ ⟨Teilzeichenkettenbegrenzung⟩ (47) ⟨Indexliste⟩ (42)

Der Name bildet sich nach der Regel:[13]

⟨Name⟩ (41, 38 | 39 | 44 | 46) : = ⟨Buchstabe⟩ (3) |

⟨Buchstabe⟩ (3) ⟨Namensteil⟩ (41a)

⟨Namensteil⟩ (41a, 41) : = ⟨Buchstabe⟩ (3) | ⟨Ziffer⟩ (5) | ⟨untergesetzter Strich⟩ |

⟨Namensteil⟩ (41a) ⟨Namensteil⟩ (41a)

Der untergesetzte Strich ist ein Sonderzeichen und damit der Gleichung 16 als weiteres übriges Sonderzeichen zuzuordnen.[14]
Korrekte Beispiele für Variablenbildung sind:

```
NAME$
I1
ATTRIBUT_NAME
KONTOBEZEICHNUNG$(L)(10:20)
Brutto(I,J)
```

Bezüglich der Namensfindung ist nach ANSI BASIC 82 die Verwendung der BASIC-Kennworte DATE, DET, ELSE, EXLINE, EXTYPE, PI, RND und TIME sowie der Kennworte DATE$ und TIME$ nicht zugelassen.[15]

13 ANSI BASIC 82, p. 18.
14 Vgl. Bd. 1, S. 74.
15 ANSI BASIC 82, p. 18. – Vgl. auch: ANSI, American National Standards Institute. Comments and Responses (Preliminary Draft). June 15, 1982. Hanover 1982, p. 8. – Im folgenden zitiert als ANSI BASIC 82 Comments.

Erwähnt sei auch, dass nach ANSI BASIC 82 eine eingebaute Funktion unterdrückt wird, sobald ihr Name in einem DIM-Statement erscheint.[16] Falls die Namensbildung gemäss ANSI BASIC 82 bei einem BASIC-Prozessor überhaupt möglich ist, sollte nach Ansicht der Verfasser generell jede Namensgleichheit mit BASIC-Kennworten oder eingebauten Funktionen vermieden werden.[17]
Zu beachten ist weiterhin, dass Variablennamen, die sich ganz oder teilweise nur in der verwendeten Buchstabenart – Gross- oder Kleinbuchstaben – unterscheiden, denselben Inhalt kennzeichnen. Dies bedeutet, dass beispielsweise die einfache numerische Variable NETTO als gleichbedeutend mit der Bezeichnung *Netto* aufgefasst wird.[18]

324 Funktionen

Bei den eingebauten numerischen[19] und alphanumerischen Funktionen[20] bestehen zwischen ANSI BASIC 82 und ANSI BASIC 79 resp. ANSI Minimal BASIC nur geringfügige Unterschiede.[21]
Bei den benutzerdefinierten Funktionen gilt für die Wahl des Funktionsnamens die gleiche Regelung wie für Variable.[22] Weiterhin lässt ANSI BASIC 82 die Verwendung von benutzerdefinierten Funktionen innerhalb arithmetischer Ausdrücke auch dann zu, wenn die Funktion nicht innerhalb des gerade laufenden BASIC-Programms definiert wurde. Es können extern benutzerdefinierte Funktionen verwendet werden, wobei deren beabsichtigte Verwendung durch ein DECLARE-Statement anzuzeigen ist.
Für die Funktion ergeben sich mithin gemäss ANSI BASIC 82 folgende Strukturgleichungen, die auch die in Band 1 erwähnten Strukturgleichungen als Sonderfall mitumfassen:[23]

16 ANSI BASIC 82, p. 19: "If an identifier which has a specified meaning in this Standard (e.g., SIN or TAB) is declared in a dimension-statement to name an array, or is declared in a function-definition to name a defined-function, then the declared interpretation of the identifier shall override the interpretation specified by this Standard."
17 Vgl. NIXDORF: Structured BASIC. Rel. 4.1. Paderborn 1981, S. 3/1–3/2. – Alphanumerische Variablenamen beginnen mit einem $-Zeichen, anstatt der bisherigen Norm zu folgen. Die Zeichenzahl darf maximal 28 sein. Als Sonderzeichen steht anstelle des untergesetzten Strichs ein Punkt.
18 ANSI BASIC 82, p. 19.
19 Vgl. Bd. 1, Abs. 325.1.
20 Vgl. Bd. 1, Abs. 325.2.
21 ANSI BASIC 82, p. 27–31 und p. 41–43.
22 ANSI BASIC 82, p. 27 und p. 41.
23 ANSI BASIC 82, p. 27 und p. 41.

⟨Benutzerdefinierte numerische Funktion⟩ 62 : = ⟨Name⟩ 41 (⟨Numerische Aktualparameterliste⟩ 63)

⟨Benutzerdefinierte alphanumerische Funktion⟩ 68 : = ⟨Name⟩ 41 $ (⟨Alphanumerische Aktualparameterliste⟩ 66)

⟨Benutzerdefinierte numerische Hybridfunktion⟩ 68e : = ⟨Name⟩ 41 (⟨Hybride alphanumerische Aktualparameterliste⟩ 68g)

⟨Benutzerdefinierte alphanumerische Hybridfunktion⟩ 68i : = ⟨Name⟩ 41 $ (⟨Hybride numerische Aktualparameterliste⟩ 68k)

Die Beschränkung der Parameteranzahl in den Aktualparameterlisten ist aufgehoben. Es muss lediglich auf exakte Übereinstimmung bezüglich Parameterzahl, bei den hybriden Parameterlisten auch bezüglich der Argumenttypen, geachtet werden.[24]
Bei den eingebauten Funktionen erscheinen die – nach ANSI BASIC 82 allerdings den Vereinbarungen zugerechneten – Funktionen SIZE, LBOUND und UBOUND besonders beachtenswert.[25] Alle drei Funktionen können mit einem oder mit zwei Argumenten aufgerufen werden. Ihre Wirkung kann der Tabelle 1 entnommen werden. Generell dienen die genannten Funktionen zur Ermittlung der minimalen oder maximalen Feldgrenzenwerte.

24 ANSI BASIC 82, p. 24 und p. 39: "The number and types of arguments in a function-ref shall correspond to the number and types of parameters specified in the definition of the function."
25 ANSI BASIC 82, p. 50.

Tabelle 1. Funktionen zur Ermittlung von Feldgrenzenwerten

Argumenten-zahl / Funktion	1 (Feldname)	2 (Feldname, Indexnummer)
SEIZE	Anzahl der Elemente im Feld	Anzahl der Elemente im n-ten Index
LBOUND	Minimal-wert des Indexes eines Vektors	Minimalwert des n-ten Indexes eines Feldes
UBOUND	Maximal-wert des Indexes eines Vektors	Maximalwert des n-ten Indexes eines Feldes

325 Vereinbarungen

Bei den Vereinbarungen hat ANSI BASIC 82 zunächst einige Erweiterungen bezüglich des DIM- und DEF-Statements zugelassen. Ausserdem sind einige neue Statements zu der Gruppe der Vereinbarungen hinzugekommen, die insbesondere das OPTION COLLATE- und DECLARE-Statement betreffen.

325.1 DIM-Statement

In DIM-Statements ist der Feldname nach den in Abs. 323 erwähnten Regeln zu bilden.[26]
Zusätzlich wurde auch die Dimensionierung erweitert, indem nach ANSI BASIC 82 die Zeilen- bzw. Spaltenzahl nicht mehr grundsätzlich von der Basis 0 oder 1 ausgehen muß, sondern die Ober- und Untergrenze der Indizes explizit festlegbar sind.

26 Vgl. Abs. 321.

Es gilt demnach:[27]

⟨Zeilenzahl⟩ : = ⟨Vorzeichenlose Ganzzahl⟩ |
89 88 24

⟨I-formatierte Ganzzahl⟩ TO ⟨I-formatierte Ganzzahl⟩
22 22

⟨Spaltenzahl⟩ : = ⟨Vorzeichenlose Ganzzahl⟩ |
90 88 24

⟨I-formatierte Ganzzahl⟩ TO ⟨I-formatierte Ganzzahl⟩
22 22

Korrekte DIM-Statements stellen demnach die folgenden Anweisungen dar:

```
100 DIM A(15,22), C$(-2 TO 17, 35)*22
110 DIM C(-1980 TO -1970), Daten(11, 12)
```

325.2 Benutzerdefinierte Funktionen

Bei der Vereinbarung von benutzerdefinierten Funktionen mittels DEF-Statement[28] hat sich nach ANSI BASIC 82 nur eine Erweiterung bezüglich des Funktionsindikators[29] ergeben, indem als Funktionsnamen allgemein ein Name zugelassen ist. Demnach gelten nunmehr folgende Gleichungen:[30]

⟨Numerischer Funktionsindikator⟩ : = ⟨Name⟩ | ⟨Name⟩ (⟨Numerische Formalparameterliste⟩)
102a 101 41 41 103

⟨Alphanumerischer Funktionsindikator⟩ : = ⟨Name⟩ $ | ⟨Name⟩ $ (⟨Alphanumerische Formalparameterliste⟩)
106 105 41 41 107

27 Vgl. Bd. 1, S. 135 und ANSI BASIC 82, p. 48–49.
28 Vgl. Bd. 1, Abs. 331.23.
29 Vgl. Bd. 1, S. 148.
30 ANSI BASIC 82, p. 72–77.

An Stelle des Funktionsnamens nach Gleichung 62a tritt nunmehr

⟨Funktionsname⟩ : = ⟨Name⟩
62a 62 | 68 41

Die bisher üblichen Funktionsnamen mit FN als Anfangsbuchstaben können weiterhin benutzt werden. Nach ANSI BASIC 82 sind die nachstehenden Beispiele korrekt.

```
100 DEF FNA(B) = SIN(DEG(B))
110 DEF Laenge(C$) = LEN(C$)
120 DEF NUMERISCH(L$)=VAL(L$)
```

Der Aufbau der extern definierten Funktionen wird in Abschnitt 333.13 erläutert, da diese Funktionen als eine spezielle Art von Unterprogrammen anzusehen sind. Ihre Verwendung innerhalb irgendwelcher Ausdrücke ist vollkommen identisch mit den durch DEF-Statements oder DEF-Blöcke definierten internen Funktionen. Es muss lediglich sichergestellt sein, dass ein Funktionsindikator nicht mehrfach definiert wird.[31] Dagegen kann ein Funktionsaufruf dazu führen, dass eine bereits aufgerufene Funktion sich selbst aufruft oder indirekt über eine andere Funktion erneut aufgerufen wird.[32]
Bei den DEF-Blöcken werden von ANSI BASIC 82 ebenfalls einige wichtige Erweiterungen eingeführt. Zunächst kann ein DEF-Block sogenannte DEF EXIT-Statements enthalten, aufgrund deren Ausführung ein DEF-Block beendet wird. Das FNEND-Statement ist nunmehr durch die Kennwortkombination END DEF ersetzt. Für den allgemeinen DEF-Block gilt mit ANSI BASIC 82:[33]

⟨DEF-Block⟩ : =

⟨DEF-Statement⟩
⟨Interner DEF-Block⟩
⟨DEFEND-Statement⟩

⟨DEF-Statement⟩ : = DEF ⟨Funktionsindikator⟩

31 Vgl. Bd. 1, S. 151; ANSI BASIC 82, p. 73.
32 ANSI BASIC 82, p. 75: "A function-definition may refer, directly or indirectly, to the function being defined; i.e., recursive function invocations are permitted."
33 ANSI BASIC 82, p. 72–73.

⟨Funktions-indikator⟩ : = ⟨Numerischer Funktions-indikator⟩ | ⟨Alphanumerischer Funktions-indikator⟩
102a 106

⟨Interner DEF-Block⟩ : = ⟨DEF let-Statement⟩ | ⟨DEFEXIT-Statement⟩ |

⟨Statement-Block⟩ | ⟨Statement-Block⟩ ⟨Interner DEF-Block⟩

⟨DEF let-Statement⟩ : = DEF ⟨Wertzuordnungsanweisung⟩
153

⟨DEFEXIT-Statement⟩ : = EXIT DEF

⟨Statement-Block⟩ : =
⟨Folge blockspezifischer BASIC-Sprachanweisungen⟩

⟨DEFEND-Statement⟩ : = END DEF

Aus diesen Strukturgleichungen ist ersichtlich, dass für die Namensbildung im Funktionsindikator allgemein ein Name angegeben werden kann. Das bisherige LET FN-Statement[34] ist durch das DEF let-Statement ersetzt.
Der Statement-Block stellt eine blockspezifische Folge von BASIC-Sprachanweisungen dar; er darf kein END-Statement[35] enthalten. Statementblöcke treten vor allem in Verbindung mit internen Blöcken insbesondere bei DEF-, FOR/NEXT-,[36] DO/LOOP-,[37] SELECT-,[38] Ausnahme-,[39] SUB-[40] und FUNCTION-Blöcken[41] auf.
Das in Band 1 angeführte Beispiel zur Verwendung des DEF-Blocks[42] ist wie folgt umzugestalten, damit es den Normvorschlägen gemäss ANSI BASIC 82 entspricht:

34 Vgl. Bd. 1, S. 151, 153, 157.
35 Zum END-Statement und seine Verwendung vgl. Bd. 1, Abs. 332.46.
36 Vgl. Abs. 329.1.
37 Vgl. Abs. 329.1.
38 Vgl. Abs. 329.1.
39 Vgl. Abs. 329.2.
40 Vgl. Bd. 2, Abs. 331.12.
41 Vgl. Abs. 333.13.
42 Vgl. Bd. 1, S. 157.

```
 10 DEF Position(A$,B$)
 20     N1=LEN(A$)
 30     N2=LEN(B$)
 40     IF N2 > N1 THEN 100
 50     FOR I = 1 TO N1-N2+1
 60        IF B$ <> A$(I,I+N2-1) THEN 90
 70        DEF Position=I
 80        DEF EXIT
 90     NEXT I
100     DEF Position=0
110 END DEF
```

325.3 Deklarationen

Nach ANSI BASIC 82 können durch spezielle Deklarationsanweisungen die prozessorspezifischen Grenzen für numerische Werte, die alphanumerische Rangordnung von Zeichen, der beabsichtigte Zugriff auf externe Funktionen und die Argumentverarbeitung der eingebauten numerischen Funktionen global für ein Programm geändert werden.
Durch das OPTION-Statement wird die numerische Genauigkeit von Zahlen und die Art der Argumentverarbeitung der eingebauten numerischen Funktionen beeinflusst. ANSI BASIC 82 sieht folgende Möglichkeiten vor:[43]

⟨OPTION-Statement⟩ : =

```
OPTION ARITHMETIC DECIMAL |
OPTION ARITHMETIC NATIVE  |
OPTION ANGLE DEGREES      |
OPTION ANGLE RADIANS
```

Durch OPTION ARITHMETIC DECIMAL wird die numerische Genauigkeit auf mindestens 10 oder mehr Stellen eingestellt, wobei für die Zahlen ein Mindestintervall von 10^{-38} bis 10^{+38} zur Verfügung stehen soll.[44] Im Fall NATIVE soll die Genauigkeit mindestens 6 Stellen umfassen. Der Mindestintervallbereich bleibt bestehen.[45] Das Statement OPTION ANGLE DEGREES schaltet die normale Argumentverarbei-

43 ANSI BASIC 82, p. 33.
44 ANSI BASIC 82, p. 33–34: "If OPTION ARITHMETIC DECIMAL is specified, or if no ARITHMETIC option is specified, then numeric-variables shall behave logically as decimal floating-point numbers, with an implementation-defined precision, say m, of at least ten significant decimal digits and with an implementation-defined range of at least 1E−38 to 1E+38."
45 ANSI BASIC 82, p. 34.

tung von geometrischen Funktionen auf die Massangabe in Grad um.[46] Ein OPTION-Statement soll in einem BASIC-Programm nur einmal vorkommen[47] und unbedingt vor dem ersten auszuführenden Statement mit einer arithmetischen Anweisung bzw. einer geometrischen Funktion stehen.[48]
Das COLLATE-Statement bestimmt die Rangordnungsart der alphanumerischen Zeichen. Es gilt:[49]

⟨COLLATE-Statement⟩ : =

COLLATE STANDARD |
COLLATE NATIVE

Im ersten Fall wird die nach ANSI X3.4-1977 (ASCII)[50] vorgeschriebene Rangordnung gewählt. Der zweite Fall legt eine prozessorspezifische Rangordnung fest. Das COLLATE-Statement soll in einem BASIC-Programm nur einmal, vor Ausführung des ersten Statements, auftreten.
Mit dem DECLARE-Statement können die implementationsabhängigen maximalen Stringlängen für Variable abgeändert[51] und zu verwendende externe Funktionen deklariert werden.[52]

46 ANSI BASIC 82, p. 34: "If OPTION ANGLE RADIANS is specified, ... then the numeric-supplied-functions COS, COT, CSC, SEC, SIN, and TAN, as well as the graphic transform-functions ROTATE and SHEAR, use arguments in radian measure, and the numeric-supplied-functions ACOS, ANGLE, ASIN, and ATN generate results in radian measure. If OPTION ANGLE DEGREES is specified, then the numeric-supplied-functions COS, COT, CSC, SEC, SIN, and TAN, as well as the graphic transform-functions ROTATE and SHEAR, use arguments in degree measure, and the numeric-supplied-functions ACOS, ANGLE, ASIN, and ATN generate results in degree measure."
47 ANSI BASIC 82, p. 33: "A program-unit shall contain at most one ARITHMETIC [ANGLE] option."
48 ANSI BASIC 82, p. 33: "An option-statement with an ARITHMETIC [ANGLE] option, if present at all, shall occur in a lower-numbered line than any numeric-expression [numeric-supplied-function or transform-function] in the same program-unit."
49 ANSI BASIC 82, p. 46. "The COLLATE option identifies the collating sequence to be used within a program-unit for comparing strings and for computing values of the CHR$ and ORD functions."
50 ANSI BASIC 82, p. 46.
51 ANSI BASIC 82, p. 46.
52 ANSI BASIC 82, p. 73–76.

⟨DECLARE-Statement⟩ : =

DECLARE STRING	⟨String-Deklaration⟩ |
DECLARE FUNCTION	⟨Funktions-Deklaration⟩

⟨String-Deklaration⟩ : = ⟨Name⟩ (41) ⟨I-formatierte Ganzzahl⟩ (22) \|

⟨String-Deklaration⟩ , ⟨String-Deklaration⟩

⟨Funktions-Deklaration⟩ : = ⟨Name⟩ (41) \| ⟨Name⟩ (41) $ ⟨Funktions-Deklaration⟩ , ⟨Funktions-deklaration⟩

Nach ANSI BASIC 82 korrekte DECLARE-Anweisungen sind

```
100 DECLARE STRING  Bezeichnung*25, ORT*35
200 DECLARE FUNCTION Externe_Funktion
```

Das DECLARE-Statement soll wie das OPTION- und COLLATE-Statement am Anfang eines Programms stehen und eine Varibale resp. eine Funktion nur einmal deklarieren.[53]

326 Eingabeanweisungen

Bei den Eingabeanweisungen wirkt sich ANSI BASIC 82 zunächst beim LINPUT-Statement[54] dahingehend aus, dass als Kennwortkombination LINE INPUT anzugeben ist. Die restliche Statementstruktur bleibt erhalten. Dasselbe gilt für das INPUT-Statement.
Die PROMPT-Identifikation kann durch eine TIMEOUT- und ELAPSED-Identifikation ergänzt werden. Dementsprechend ist in den – auf ANSI BASIC 79 basierenden – Gleichungen 117a und 117c ⟨PROMPT-

53 ANSI BASIC 82, p. 46 und p. 74.
54 Vgl. Bd. 1, S. 169.

Identifikation⟩ im Sinne von ANSI BASIC 82 durch ⟨Identifikation⟩ zu ersetzen.[55] Dabei gilt:[56]

⟨Identifikation⟩ : = ⟨PROMPT-Identifikation⟩
117b

⟨TIMEOUT-Identifikation⟩ |
⟨ELAPSED-Identifikation⟩ |
⟨Identifikation⟩ , ⟨Identifikation⟩

⟨TIMEOUT-Identifikation⟩ : = TIMEOUT ⟨Arithmetischer Ausdruck⟩
49

⟨ELAPSED-Identifikation⟩ : = ELAPSED ⟨Arithmetischer Ausdruck⟩
49

In einem INPUT oder LINE INPUT-Statement können alle drei Identifikationen gleichzeitig auftreten. Die TIMEOUT-Identifikation führt dazu, dass die Variablenliste innerhalb der mit dem arithmetischen Ausdruck in Sekunden angegebenen Zeit aufgefüllt sein muss, andernfalls soll nach ANSI BASIC eine Ausnahmeregelung eintreten.[57] Dies etwa in dem Sinne, dass mit dem Programmablauf fortgefahren wird und der Anwender die unvollständige Dateneingabe selbst kontrollieren muss. Durch das Statement

```
130 INPUT TIMEOUT 15: A$
```

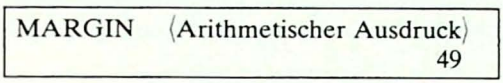
wird 15 Sekunden Realzeit auf die Eingabe gewartet und andernfalls die Ausnahmeregelung durchgeführt.

55 Vgl. Bd. 1, S. 168–169.
56 ANSI BASIC 82, p. 87–90.
57 ANSI BASIC 82, p. 88: "If no valid input-reply or line-input-reply has been supplied within S seconds, then an exception shall occur."

327 Ausgabeanweisungen

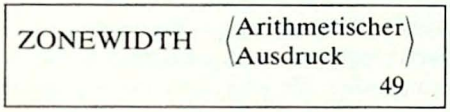
Die Ausgabeanweisungen bleiben nach ANSI BASIC 82 in ihrer Struktur vollkommen erhalten; sie werden durch das MARGIN- und ZONWIDTH-Statement ergänzt.[58]
Für das MARGIN-Statement gilt:[59]

⟨MARGIN-Statement⟩ : =



Der arithmetische Ausdruck wird ganzzahlig gemacht und gibt dann die maximale Zeichenzahl pro Zeile an. Der Wert sollte grösser als 15 sein. Das ZONEWIDTH-Statement erlaubt die Standardbreite der Druckzone von 15 auf einen beliebigen Wert zu ändern.[60]

⟨ZONEWIDTH-Statement⟩ : =



328 Programmsteuerungsanweisungen

Die neuen Softwaretechniken der blockstrukturierten Programmierung werden von ANSI BASIC 82 in vollem Umfang berücksichtigt.[61] Damit verbunden sind Änderungen und Erweiterungen in den Programmsteuerungsanweisungen, die vor allem eine einfache Programmierung der verschiedenen Wiederholungsblöcke[62], des Auswahlblocks[63] und eines neu entwickelten Ausnahmeblocks zulassen.[64]

58 ANSI BASIC 82, p. 91.
59 ANSI BASIC 82, p. 94: "Execution of a margin-statement shall cause its index to be evaluated and to become the new margin."
60 ANSI BASIC 82, p. 94: "Execution of a zone-width-statement shall cause its index to be evaluated and to become the new zone width."
61 Vgl. hierzu auch Kapitel 2.
62 Vgl. Abs. 22.
63 Vgl. Abs. 22.
64 Vgl. Abs. 329.3.

Im Zusammenhang mit den Erweiterungen der Programmsteuerungsanweisungen und auch der Lokalisierung von Vereinbarungen in einem BASIC-Programm hat Roger D. Ross (NCR Corporation) in der Diskussion um zukünftige ANSI BASIC-Standards die Frage der »Implementation conformance by BASIC interpreters« aufgeworfen und darauf hingewiesen, dass zur Bearbeitung derartig hochentwickelter Anweisungen einfache Interpreter nicht mehr ausreichen und nunmehr zur Übersetzung derartiger Anweisungsformen Compiler[65] notwendig sind.[66] Die ANSI BASIC-Commission erkennt diese Notwendigkeit noch nicht an und lehnt die Präferenzierung einer speziellen Implementationsmethode für BASIC weiterhin ab.[67]

328.1 Wiederholungsblöcke

Wiederholungsblöcke sind BASIC-Programmabschnitte, die zyklisch durchlaufen werden, wobei die Kontrolle des Zyklenablaufs über eine Variable durchgeführt wird (FOR/NEXT-Block) oder durch Abbruchbedingungen am Blockanfang, Blockende oder im Block geschieht (DO/LOOP-Block). Alle Formen dieser Wiederholungs-Blöcke konnten bisher durch die bereits in ANSI BASIC 79 festgelegten Elementaranweisungen realisiert werden.[68] Teilweise mussten dazu allerdings mehrere Elementaranweisungen angegeben werden, die jetzt in einem Statement zusammengefasst sind.

Der FOR/NEXT-Block bleibt bezüglich des FOR- und NEXT-Statements vollkommen erhalten.[69] Für die interne Struktur des FOR/NEXT-Blocks wurde eine GOTO-unabhängige Abbruchanweisung durch das EXIT FOR-Statement geschaffen. Mit dieser Anweisung kann eine

65 Zum Begriff des Interpreters und Compilers vgl. Weber und Türschmann. FOSBIC-Compiler. Ein BASIC-Compiler auf FORTRAN-Basis. Bern und Stuttgart 1977, S. 33–34.

66 ANSI BASIC 82 Comments, p. 1: "Several specific features, as described in the sementics section, preclude their use in an interpretive environment. For example, both the DECLARE and DIM statements state that when execution encounters these statements, it shall proceed to the next line with no effect. Another example is that no statement outside a for-loop or do-loop shall refer to a line in a for-body or loop-body. Both of these descriptions infer some sort for pre-scanning mechanism found in compilers, but not in interpreters."

67 ANSI BASIC 82 Comments, p. 2: "While there has never been any intent to show a preference for any specific implementation method."

68 Vgl. Abs. 22.

69 Vgl. Bd. 1, Abs. 322.43 und ANSI BASIC 82, p. 65.

FOR/NEXT-Programmschleife jederzeit abgebrochen werden, ohne dass die Schleifenvariable[70] die Laufbereichsgrenze[71] erreicht haben muss. Für die Struktur des FOR/NEXT-Blocks gilt nach ANSI BASIC 82:[72]

⟨FOR/NEXT-Block⟩ : =

⟨FOR-Statement⟩
176

⟨Interner FOR/NEXT-Block⟩

⟨NEXT-Statement⟩
177

⟨Interner FOR/NEXT-Block⟩ : = ⟨FOREXIT-Statement⟩ |
⟨Statement-Block⟩ | ⟨Statement-Block⟩ |
⟨Interner FOR/NEXT-Block⟩

⟨FOREXIT-Statement⟩ : = EXIT FOR

Die Eingabe von Textzeilen in einen Vektor wird in dem folgenden FOR/NEXT-Block abgebrochen, sobald eine Zeile ohne Text eingegeben wird.

```
1000 FOR I=1 TO N
1010     INPUT A$(I)
1020     IF LEN(A$(I))=0 THEN EXIT FOR
1030 NEXT I
```

Der nach ANSI BASIC 79 vorgesehene LOOP/ENDLOOP-Block[73] wird nach ANSI BASIC 82 in einen DO/LOOP-Block umbenannt. Der Start dieses zyklischen Blocks erfolgt durch ein DO-Statement; das Blockende wird durch ein LOOP-Statement in neuer Form angezeigt. Zur Struktur dieses DO/LOOP-Blocks kann folgendes angegeben werden:[74]

70 Zur Schleifenvariablen vgl. Bd. 1, S. 225.
71 Zur Laufbereichsgrenze vgl. Bd. 1, S. 225.
72 ANSI BASIC 82, p. 65.
73 Vgl. Bd. 1, Abs. 322.44.
74 ANSI BASIC 82, p. 65–67.

⟨DO/LOOP-Block⟩ : =

⟨DO-Statement⟩

⟨Interner DO/LOOP-Block⟩

⟨LOOP-Statement⟩

⟨DO-Statement⟩ : = DO ⟨Abbruchbedingung⟩ | DO

⟨LOOP-Statement⟩ : = LOOP ⟨Abbruchbedingung⟩ | LOOP

⟨Abbruchbedingung⟩ : = UNTIL ⟨Logisch-komparativer Ausdruck⟩ | 52

WHILE ⟨Logisch-komparativer Ausdruck⟩ 52

⟨Interner DO/LOOP-Block⟩ : = ⟨DOEXIT-Statement⟩

⟨Statement-Block⟩ | ⟨Statement-Block⟩ ⟨Interner DO/LOOP-Block⟩

⟨DOEXIT-Statement⟩ : = EXIT DO

Realisiert man das Beispiel zum FOR/NEXT-Block mit einem DO/LOOP-Block, so sind folgende Lösungen möglich:

```
1000 I=1
1010 DO UNTIL I > N
1020    INPUT A$(I)
1030    IF LEN(A$(I))=0 THEN EXIT DO
1040    I=I+1
1050 LOOP
```

```
1000 I=1
1010 DO
1020    INPUT A$(I)
1030    IF LEN(A$(I))=0 THEN EXIT DO
1040    I=I+1
1050 LOOP UNTIL I>N
```

```
1000 I=1
1010 DO WHILE I <= N
1020    INPUT A$(I)
1030    IF LEN(A$(I))=0 THEN EXIT DO
1040    I=I+1
1050 LOOP
```

```
1000 I=1
1010 DO
1020    INPUT A$(I)
1030    IF LEN(A$(I))=0 THEN EXIT DO
1040    I=I+1
1050 LOOP WHILE I <= N
```

Aus diesen Beispielen ist deutlich die Wirkung der Abbruchbedingung mit UNTIL und WHILE ersichtlich. Ein DO/LOOP-Block wird dann beendet, wenn die UNTIL-Abbruchbedingung „wahr" ist oder wenn die WHILE-Abbruchbedingung „falsch" ist.[75]
Kritik an dieser neuen Norm des DO/LOOP-Blocks ist insofern zu üben, als das bisherige Startkennwort LOOP nunmehr als Blockendekennwort erscheint und dadurch erhebliche Kompatibilitätsprobleme zwischen ANSI BASIC 79 und ANSI BASIC 82 entstehen.[76]
Eine interessante Ergänzung des internen FOR/NEXT-Blocks sowie des internen DO/LOOP-Blocks stellen das NEXT FOR- und das NEXT LOOP-Statement dar. Diese Statements finden sich bereits in einigen BASIC-Dialekten; bei der FOSBIC Version 2 sind sie ebenfalls in der Form

⟨NEXT FOR-Statement⟩ := NEXT FOR

⟨NEXT LOOP-Statement⟩ := NEXT LOOP

verfügbar. Beide Statements unterbrechen den Programmablauf und bewirken die Programmfortsetzung mit dem NEXT- bzw. LOOP-Statement am Blockende. Damit kann ohne GOTO-Statement direkt auf das Blockende verzweigt bzw. ein Block ohne IF-/ELSE-Statementkombination auf einfache Weise in Blockteile zerlegt werden.

328.2 Auswahlblöcke

Während bisher ein Auswahlblock durch eine Folge von IF-Blöcken[77] realisiert wurde[78], normiert ANSI BASIC 82 hierzu die Statementkombination des SELECT-Blocks. Seine Struktur wird wie folgt vorgegeben:[79]

75 ANSI BASIC 82, p. 66: "An exit-condition shall be said to require exit from a loop if the value of the relational-expression following the keyword WHILE is false or if the value of the relational-expression following the keyword UNTIL is true."
76 Vgl. hierzu auch ANSI BASIC 82 Comments, p. 49–50, wo für die Verwendung von END DO anstelle von LOOP plädiert wird; die entsprechenden Vorschläge führten bisher allerdings zu keiner Abänderung von ANSI BASIC 82.
77 Vgl. Bd. 1, Abs. 332.422.
78 Vgl. Abs. 22.
79 ANSI BASIC 82, p. 68–71.

⟨SELECT/END SELECT-Block⟩ : =

⟨SELECT-Statement⟩

⟨Interner SELECT-Block⟩

⟨SELECT END-Block⟩

⟨SELECT-Statement⟩ : = SELECT ⟨Variable⟩
35

⟨Interner SELECT-Block⟩ : = ⟨CASE-Block⟩ | ⟨CASE-Block⟩
⟨Interner SELECT-Block⟩

⟨CASE-Block⟩ : = ⟨CASE-Statement⟩ ⟨Statement-Block⟩

⟨CASE-Statement⟩ : = CASE ⟨Auswahlliste⟩ | CASE ELSE

⟨Auswahlliste⟩ : = IS ⟨Vergleichsoperator⟩ ⟨Konstante⟩ |
13 17

⟨Konstante⟩ TO ⟨Konstante⟩ |
17 17

⟨Konstante⟩ |
17

⟨Auswahlliste⟩ , ⟨Auswahlliste⟩

⟨SELECT END-Statement⟩ : = END SELECT

Durch die Anweisungsfolge

```
10 SELECT A$(1:1)
20 CASE "A" TO "Z", "a" TO "z"
30    PRINT A$;" starts with a letter !!!"
40 CASE "0" TO "9"
50    PRINT A$;" starts with a digit !!!"
60 CASE ELSE
70    PRINT A$;" doesnot start with a letter or a digit !!!"
80 END SELECT
```

wird in der alphanumerischen Variablen A$ bestimmt, mit welchen Zeichen der Textinhalt beginnt.
Auch das nächste Beispiel zeigt, dass in der Auswahlliste des CASE-Statements nur Konstante auftreten dürfen.

```
100 SELECT X
110 CASE IS < 0
120   PRINT X; " IS NEGATIVE"
130 CASE IS > 0
140   PRINT X; " IS POSITIVE"
150 CASE ELSE
160   PRINT X; " IS ZERO"
170 END SELECT
```

Vermerkt sei, dass es nach ANSI BASIC 82 nicht untersagt ist, in einem CASE-Block das GOTO-Statement zu verwenden.[80] Damit kann ein SELECT-Block direkt verlassen und ein anderes BASIC-Statement ausgeführt werden.
Allgemein ist jedoch zu bemerken, dass es grundsätzlich nicht erlaubt ist, durch ein GOTO-Statement von ausserhalb in einen Block hineinzuspringen bzw. von einem CASE-Block in einen anderen zu springen.
Weiterentwicklungen des Ausnahmeblocks erlauben, dass anstelle der Variablen im SELECT-Statement und anstelle der Konstanten in der Auswahlliste des CASE-Statements allgemeine Ausdrücke eingesetzt werden dürfen. Die FOSBIC Version 2 ist bereits auf diese Möglichkeit erweitert.

328.3 Ausnahmeblöcke

Neben den in Abschnitt 335 behandelten Diagnoseanweisungen gestattet ANSI BASIC 82 die Programmierung sogenannter Ausnahmeblöcke. Gegenüber den Diagnoseanweisungen sind die Ausnahmeblöcke fester Bestandteil eines BASIC-Programms und dienen nicht dazu, den Programmtest zu erleichtern, sondern sie sollen vielmehr dazu eingesetzt werden, Programmfehler abzufangen, die durch falsche Programmanwendung entstehen können.
Die Struktur des Ausnahmeblocks regelt ANSI BASIC 82 wie folgt:[81]

⟨Ausnahmeblock⟩ : =

⟨WHEN-Statement⟩
  ⟨Statement-Block⟩

⟨USE-Statement⟩
  ⟨Interner USE-Block⟩

⟨WHENEND-Statement⟩

80 ANSI BASIC 82, p. 13–14.
81 ANSI BASIC 82, p. 150–153.

⟨WHEN-Statement⟩ : = WHEN EXCEPTION IN

⟨USE-Statement⟩ : = USE

⟨Interner USE-Block⟩ : = ⟨Ausnahme-Statement⟩ |
⟨Statement⟩ |
⟨Statement⟩ ⟨Interner USE-Block⟩

⟨Ausnahme-Statement⟩ : = ⟨RETRY-Statement⟩ |
⟨CONTINUE-Statement⟩ |
⟨WHENEXIT-Statement⟩

⟨RETRY-Statement⟩ : = RETRY

⟨CONTINUE-Statement⟩ : = CONTINUE

⟨WHENEXIT-Statement⟩ : = EXIT HANDLER

⟨WHEN END-Statement⟩ : = END WHEN

Die Ablauffolge des Ausnahmeblocks sieht vor, dass normalerweise der dem WHEN-Statement folgende Statement-Block ausgeführt wird. Sobald in der Ausführung dieses Blocks ein Programmfehler auftritt, geht der Programmablauf auf die Statementfolge des internen USE-Blocks über, die dem USE-Statement nachsteht.
In diesem internen USE-Block können die oben erwähnten Ausnahme-Statements enthalten sein. Das RETRY-Statement verursacht, dass der Programmablauf auf dasjenige Statement des normalen Statement-Blocks zurückgeht, das die Ausnahme verursacht hat; vgl. hierzu das folgende Beispiel.

```
100 WHEN EXCEPTION IN
110   PRINT "BITTE GEBEN SIE IHR ALTER EIN"
120   INPUT A
130 USE
140   PRINT "BITTE NUR ZAHLENWERTE EINGEBEN"
150   RETRY
160 END WHEN
```

In dem obigen Beispiel wird durch den Ausnahmeblock die versehentliche Eingabemöglichkeit eines Textes anstatt einer Zahl abgefangen und die Eingabe solange wiederholt, bis sie korrekt ist.

Das CONTINUE-Statement führt den Programmablauf mit dem Statement fort, das demjenigen Statement folgt, das die Ausnahme verursachte.
Durch das WHENEXIT-Statement wird er Programmablauf mit der dem END WHEN-Statement folgenden Anweisung fortgesetzt.
Neben den Ausnahme-Statements erlaubt ANSI BASIC 82 im internen USE-Block den Aufruf der beiden parameterlosen, eingebauten Funktionen EXLINE und EXTYP. Die eingebaute Funktion EXLINE soll dabei den Wert der Statementnummer des Statements übergeben, das die Ausnahme erzeugt. Mit EXTYP wird ein Zahlenwert übergegeben, der die Art der erzeugten Ausnahme (Fehlercode) angibt.

33 Sonderanweisungen

Die Sonderanweisungen werden in Matrizen-, File-, erweiterte Programmsteuerungs- und Graphikanweisungen unterteilt; vgl. Abbildung 1, Bd. 2.

331 Matrizenanweisungen

Die bei den Matrizenanweisungen durch ANSI BASIC 82 vorgenommenen Änderungen betreffen zunächst nur die Bildung der Feldnamen. Allgemein kann in allen Matrizenanweisungen der Feldname nunmehr durch einen Namen ersetzt werden, der sich aus bis zu 31 Zeichen bildet.[82] Es gilt nun:[83]

⟨Feldname⟩ : = ⟨Name⟩
40

Ein zulässiges Matrizenstatement ist die Anweisung:

```
130 MAT BRUTTO = NETTO + MEHRWERTSTEUER
```

unter der Annahme, dass Brutto, Netto und Mehrwertsteuer Felder gleicher Dimensionsgrössen sind.
Während in den bisherigen MAT-Anweisungen keine Vereinbarungen existieren,[84] kann das mit ANSI BASIC 82 eingeführte REDIM-State-

82 Vgl. Abs. 323.
83 ANSI BASIC 82, p. 52–58.
84 Vgl. Bd. 2, Abs. 33.

ment[85] dieser Anweisungsgruppe zugeordnet werden. Seine Struktur unterliegt der nachstehenden Regel:[86]

⟨REDIM-Statement⟩ : =

REDIM ⟨Redeklarationsliste⟩

Die Redeklarationsliste unterscheidet sich von der im DIM-Statement[87] angebbaren Deklaration nur dadurch, dass bei den einzelnen Deklarationen anstelle der sonst zwingend vorgeschriebenen positiven Ganzzahl arithmetische Ausdrücke erscheinen dürfen.

⟨Redeklarationsliste⟩ : = ⟨Redeklaration⟩ |

⟨Redeklaration⟩ , ⟨Redeklarationsliste⟩

⟨Redeklaration⟩ : = ⟨Name⟩ (⟨Redimensionierung⟩) |
41

⟨Name⟩ $ (⟨Redimensionierung⟩)
41

⟨Redimensionierung⟩ : = ⟨Redimensionierte Zeilenzahl⟩ | ⟨Redimensionierte Spaltenzahl⟩ |

⟨Redimensionierte Zeilenzahl⟩ , ⟨Redimensionierte Spaltenzahl⟩

⟨Redimensionierte Zeilenzahl⟩ : = ⟨Arithmetischer Ausdruck⟩ | TO ⟨Arithmetischer Ausdruck 49⟩ |
49

⟨Arithmetischer Ausdruck 49⟩ TO ⟨Arithmetischer Ausdruck 49⟩

85 ANSI BASIC 82, p. 48 und 51. – Nach ANSI BASIC 82 Comments, p. 38 wird das REDIM-Statement aus ANSI BASIC 82 eliminiert; zur Diskussion vgl. ANSI BASIC 82 Comments, p. 35–38.
86 ANSI BASIC 82, p. 49.
87 Vgl. Bd. 1, Abs. 331.21.

⟨Redimensionierte Spaltenzahl⟩ : = ⟨Arithmetischer Ausdruck⟩ 49 | TO ⟨Arithmetischer Ausdruck 49⟩ |

⟨Arithmetischer Ausdruck 49⟩ TO ⟨Arithmetischer Ausdruck 49⟩

Die arithmetischen Ausdrücke einer Redimensionierung werden ganzzahlig gemacht und anschliessend die Feldgrenzen des angegebenen Feldes auf die berechneten Grenzwerte umgesetzt. Die Anzahl der nach der Redimensionierung beanspruchten Felder darf nicht grösser sein als die mit der Dimensionierung maximal festgelegte Anzahl.[88] Der Vorteil des REDIM-Statements gegenüber den sonstigen Möglichkeiten, die Dimensionsgrenzen mit MAT-Anweisungen zu verändern, besteht darin, dass mit dem REDIM-Statement auch die Felduntergrenzen beeinflusst werden können. Alle anderen MAT-Anweisungen erlauben nur die Veränderung der Feldobergrenzen.
Fehlt bei einer Redimensionierung der Zeilen- oder Spaltenzahl der linke Arithmetische Ausdruck in einer TO-Angabe, so bleibt die untere Feldgrenze erhalten; das REDIM-Statement entspricht in seiner Wirkung dann einem MAT let-Statement mit der Funktion ZER.[89] Die redimensionierte obere Feldgrenze darf in diesem Fall die Untergrenze nicht unterschreiten.
Für die zukünftige Entwicklung des REDIM-Statements bezüglich alphanumerischer Felder ist zu erwarten, dass neben der Redimensionierung der Feldgrenzen auch eine Änderung der Zeichenzahl je Element durch einen Stringlängentrimmer zugelassen wird.[90] In diesem Fall sind als Redeklaration dann die Angaben

⟨Redeklaration⟩ : = ⟨Name⟩ 41 ((⟨Redimensionierung⟩)) |

⟨Name⟩ 41 $ ((⟨Redimensionierung⟩)) |

⟨Name⟩ 41 $ ((⟨Redimensionierung⟩)) ⟨Stringlängen-trimmer⟩ 223

88 ANSI BASIC 82, p. 51: "The total number of elements for the redimensioned array shall not exceed the number of elements specified in the DIM statement for that array."
89 Vgl. Bd. 2, Abs. 331.221.
90 Zum Stringlängen-Trimmer vgl. Bd. 2, Abs. 331.222.

zulässig. Diese Form ist zwar nach ANSI BASIC 82 noch nicht vorgesehen; sie lässt aber aus dem Entwicklungstrend von ANSI BASIC 79 zu 82 ableiten.
Korrekte REDIM-Anweisungen nach ANSI BASIC 82 stellen die folgenden Statements dar:

```
150 REDIM FELD X(-25 TO 25, 37)
250 REDIM NAME$(23,88), A1$(TO 20, TO 30)
```

Beim MAT INPUT- und MAT LINPUT-Statement ergeben sich ebenfalls geringfügige Änderungen bzw. Ergänzungen.
Das Kennwort des MAT LINPUT-Statements wird auf MAT LINE INPUT abgeändert.[91] Anstelle der PROMPT-Identifikation können nun auch die TIMEOUT- und ELAPSED-Identifikationen treten.[92] Die neuen Strukturgleichungen lauten wie folgt:[93]

⟨MAT INPUT-Statement⟩ : =
227 209

MAT INPUT ⟨MAT Inputliste⟩ |
227a

MAT INPUT ⟨Identifikation ⟩ : ⟨MAT Inputliste⟩
227a

⟨MAT LINE INPUT-Statement⟩ : =
227c 209

MAT LINE INPUT ⟨MAT Linputliste⟩ |
227d

MAT LINE INPUT ⟨Identifikation ⟩ : ⟨MAT Linputliste⟩
227d

91 ANSI BASIC 82, p. 101.
92 ANSI BASIC 82, p. 101 und p. 102: "The input-modifier-list, if present, shall work as described in section 10.2 [elementares Input]".
93 ANSI BASIC 82, p. 101; vgl. auch Bd. 2, Abs. 331.12.

Der Aufbau der Identifikation entspricht vollständig dem des elementaren INPUT- bzw. LINE INPUT-Statements; vgl. Abs. 327.
Nach dem neuen Standard wären demnach korrekte MAT INPUT-Anweisungen:

```
230 MAT INPUT PROMPT "KOEFFIZIENTENMATRIX EINGEBEN": A
240 MAT LINE INPUT TIMEOUT 100, PROMPT "EINGABEZEIT MAX. 100 SEC.": C
```

Bei den MAT-Wertzuordnungsanweisungen sollen nach ANSI BASIC 82 bei Verwendung von arithmetischen Ausdrücken im numerischen MAT-Ausdruck[94] die bisher vorgeschriebene rechte und linke Klammer entfallen.[95] Dies dürfte die bisher realisierte Klarheit des MAT-Ausdrucks stören; es wird deshalb empfohlen, den bisherigen Klammergebrauch beizubehalten.
Bei den MAT-Ausgabeanweisungen treten keine Änderungen gegenüber ANSI BASIC 79 auf.[96]
Erwähnt sei, dass der in Band 2, Gleichung 223, definierte Stringlängentrimmer nach ANSI BASIC 82 in allen Matrizenanweisungen, in denen er auftreten kann, zum Standard erhoben wird.

332 Fileanweisungen

Die nach ANSI BASIC 82 vorgesehenen Fileanweisungen weichen teilweise fundamental von ANSI BASIC 79 ab.[97] Beide Normvorschläge können seitens der Computerhersteller als Bezugsrahmen zur Weiterentwicklung ihrer BASIC-Prozessoren herangezogen werden.
In den folgenden Abschnitten werden die wesentlichen Besonderheiten von ANSI BASIC 82 herausgearbeitet, wobei neue Statementformen in möglichst enger Verbindung zu den Ausführungen in Band 2 diskutiert werden.[98]
Dem Leser wird empfohlen, den Abstimmungsgrad der von ihm aktuell verwendeten BASIC-Version mit ANSI BASIC 79 resp. ANSI BASIC 82 minutiös zu ermitteln und erkennbare Entwicklungstendenzen zu vermerken.
Es ist zu erwarten, dass sich ANSI BASIC 82 bei den Computerherstel-

94 Vgl. Bd. 2, Abs. 331.211 und dort Gleichung 232.
95 ANSI BASIC 82, p. 52.
96 ANSI BASIC 79, p. 101; vgl. auch Bd. 2, Abs. 331.3.
97 Vgl. ANSI BASIC 79, p. 54–60; ANSI BASIC 82, p. 106–149.
98 Vgl. Bd. 2, Abs. 332.

lern nur langsam und kaum vollständig durchsetzen wird.[99] Dagegen gilt eine praktische unveränderte Übernahme von ANSI BASIC 82 durch die ECMA (European Computer Manufactures Association) als sicher.[100] Als einschneidenste Ergänzung bei den Fileanweisungen führt ANSI BASIC 82 den neuen Typ der Filemodifikationsanweisungen ein.[101] Die Fileanweisungen gliedern sich nach Gleichung 258 nunmehr wie folgt:

⟨Fileanweisung⟩ : = ⟨FL-Vereinbarung⟩ |
258 206 259

⟨FL-Ausgabeanweisung⟩ |
273

⟨FL-Modifikationsanweisung⟩ |

⟨FL-Eingabeanweisung⟩ |
283

⟨FL-Kontrollanweisung⟩ |
288

§32.1 Organisation und Struktur von Files

Files bestehen auch nach ANSI BASIC 82 aus einer Menge von Sätzen (records). Im Grenzfall kann ein Record nur einen Wert umfassen, so dass ein File diesfalls lediglich aus einer Menge von Einzelwerten aufgebaut ist.[102]

99 Vgl. hierzu ANSI BASIC 82 Comments, p. 69–70, insbesondere: "The current files section has been evolved in response to a perceived need by users and implementors and reflects a long series of committee votes and decisions. In particular, business users need to be able to process COBOL-style files. It was recognized that the more elaborate file types might represent for many users an unneeded and unwanted facility. For this reason, such types were put into optional modules. The core contains only such files as seemed commonly useful. Core files are implementible using sequential devices (e. g., tape) only (as opposed to disk)."

100 Brief der ECMA (L. Lauri, Technical Officer) vom 19. Juli 1982: "As per your question re ECMA following the ANSI Standard I can only answer that the ECMA Standard, when published will follow the ANSI Standard in all the part that will be incorporated in it. For the file statement, ECMA has donė a lot of work on it and will certainly adopt it. The only differences between the ANSI and ECMA Standards will be editorials (layout, introduction, references)."

101 Vgl. Abs. 332.4.

102 ANSI BASIC 82, p. 106: "Files are organized collections of data external to BASIC programs."

Der inhaltliche Aufbau der Sätze bestimmt die Filestruktur, ihr logischer Aufbau legt die Fileorganisation fest.[103]

332.11 Filestruktur

ANSI BASIC 82 unterscheidet drei Satztypen, die als zeichen-, inhalt- oder feldbezogen definiert und durch die Kennworte DISPLAY, INTERNAL resp. NATIVE spezifiziert werden.[104] Dementsprechend kann zwischen

- zeichenstrukturierten Files
- inhaltstrukturierten Files und
- feldstrukturierten Files

differenziert werden.
Zeichenstrukturierte Files sind aus DISPLAY Records[105] aufgebaut; inhaltstrukturierte Files bestehen aus INTERNAL Records.[106] Beide Filetypen sind auch nach ANSI BASIC 79 vorgesehen.[107]
Die feldstrukturierten Files werden aus NATIVE Records aufgebaut, die ihrerseits aus einer Folge beliebig strukturierter Felder bestehen.[108] Die Felder können zur Aufnahme von Zeichenfolgen oder Inhaltsmengen dienen; ihre Festlegung erfolgt durch das TEMPLATE-Statement.[109]
Das TEMPLATE-Statement entspricht in seiner Wirkungsweise einem IMAGE-Statement.[110] Nach ANSI BASIC 82 gilt:

⟨TEMPLATE-Statement⟩ : =

TEMPLATE : ⟨Schablonenliste⟩

103 Vgl. Bd. 2, Abs. 332.1.
104 ANSI BASIC 82, p. 106: "There are three kinds of record types: display, internal, and native."
105 ANSI BASIC 82, p. 106: "A display record is a sequence of characters."
106 ANSI BASIC 82, p. 106: "An internal record is a sequence of typed values."
107 Vgl. Bd. 2, Abs. 332.111 und 332.112.
108 ANSI BASIC 82, p. 106: "A native record is a sequence of fields, as described by a program-specified template."
109 ANSI BASIC 82, p. 117: "[The] TEMPLATE, in conjunction with the list of operands of the associated record operation, specifies the size, type, number, and order of fields within the record."
110 Vgl. ANSI BASIC 82, p. 126–127. – Zum IMAGE-Statement vgl. Bd. 1, Abs. 332.23.

Zu beachten ist aber, dass sich aus der Struktur der Schablonen keinerlei Verbindungen zur Image-Maske des IMAGE-Statements ergeben.[111]

⟨Schablonenliste⟩ : = ⟨Schablone⟩ | ⟨Schablone⟩ , ⟨Schablonenliste⟩

$$\langle \text{Schablone} \rangle : = \left\langle \begin{matrix} \text{Numerische} \\ \text{Schablone} \end{matrix} \right\rangle \mid \left\langle \begin{matrix} \text{Alphanumerische} \\ \text{Schablone} \end{matrix} \right\rangle$$

$$\left\langle \begin{matrix} \text{Numerische} \\ \text{Schablone} \end{matrix} \right\rangle : = \langle \text{Feldzähler} \rangle \text{ OF NUMERIC } * \langle \text{Numerisches Format} \rangle$$

$$\langle \text{Feldzähler} \rangle : = \langle \text{SKIP} \rangle \mid ? \mid \underset{22}{\langle \text{I-formatierte Ganzzahl} > 0 \rangle}$$

$$\langle \text{Numerisches Format} \rangle : = \text{E} \mid \langle \text{I-formatierte Ganzzahl} \rangle \mid \underset{27}{\langle \text{F-formatierte Gleitkommazahl} \rangle}$$

$$\left\langle \begin{matrix} \text{Alphanumerische} \\ \text{Schablone} \end{matrix} \right\rangle : = \langle \text{Feldzähler} \rangle \text{ OF STRING } * \underset{22}{\left\langle \begin{matrix} \text{I-formatierte} > 0 \\ \text{Ganzzahl} \end{matrix} \right\rangle}$$

Die TEMPLATE-Anweisung

```
200 TEMPLATE: 3 OF NUMERIC * 6.2, ? OF STRING *27
```

ist so zu interpretieren, dass eine das TEMPLATE-Statement 200 benutzende Fileanweisung 3 numerische Felder mit F-formatierten Gleitkommazahlen (mit insgesamt 6 Stellen, bei 2 Nachkommastellen) vor einer unbekannten Zahl von alphanumerischen Feldern (mit jeweils 27 Zeichen) bearbeitet. Bei der Angabe E in der numerischen Schablone werden E-formierte Zahlen erwartet.[112]

332.12 Fileorganisation

Nach ANSI BASIC 82 sind für den logischen Aufbau der Sätze innerhalb eines Files vier verschiedene Möglichkeiten vorgesehen, die mittels der

111 Vgl. Bd. 1, S. 187–188.

112 Vgl. hierzu auch die Beispiele bei ANSI BASIC 82, p. 129. – Zur Formatierung vgl. Bd. 1, Abs. 322.

Kennworte SEQUENTIAL, STREAM, RELATIVE und KEYED festgelegt werden.[113] Im folgenden wird deshalb zwischen

- sequentieller Fileorganisation
- fortlaufender Fileorganisation
- relativer Fileorganisation und
- indizierter Fileorganisation

unterschieden.
Die sequentielle Fileorganisation (sequential file) stimmt gemäss ANSI BASIC 82 – bis auf kennwortspezifische Unterschiede – mit den Vorschlägen von ANSI BASIC 79 überein.[114]
Die relative Fileorganisation (relative file) wird in der Form normiert, wie sie in Band 2 dargestellt ist.[115] Unterschiede finden sich in einigen Kennworten und im syntaktischen Aufbau der Adresse.[116]
Die fortlaufende Fileorganisation (stream file) ist eine spezielle Form der sequentiellen Fileorganisation, indem hier das File nur aus einem einzigen Satz und dieser aus einer Folge von Inhalten besteht.[117]
Da die fortlaufende Fileorganisation als Sonderfall der sequentiellen Fileorganisation aufgefasst werden kann, wird sie im folgenden nicht eingehender behandelt. Es ist auch zu erwarten, dass die fortlaufende Fileorganisation bei zukünftigen Normvorschlägen als eigenständige Fileorganisationsform wegfallen wird.
Die indizierte Fileorganisation (keyed file) ist als echte Weiterentwicklung der Fileorganisation anzusprechen. Für diese Organisationsform ist charakteristisch, dass ein File aus einer Folge von Sätzen besteht, wobei Sätze nunmehr durch einen Schlüssel (key) identifiziert werden.[118] Die lo-

113 ANSI BASIC 82, p. 115: "The file-organisation specifies the logical relationship between file elements, the nature of those elements, and the means by which the file pointer can be manipulated to identify the elements. The organization is specified with one of the keywords SEQUENTIAL, STREAM, RELATIVE, or KEYED, or with a string-expression whose value is one of these keywords."

114 ANSI BASIC 82, p. 115: "A sequential file is a sequence of records. The order of the records is established by the order in which they were written." – Vgl. auch Bd. 2, Abs. 332.121.

115 ANSI BASIC 82, p. 116: "A relative file is a sequence of record-areas, each of which may or may not contain a record. The record-areas are numberd sequentially beginning with 1." – Vgl. auch Bd. 2, Abs. 332.122.

116 Zur Adresse vgl. Bd. 2, Abs. 332.31 und Abs. 332.31.

117 ANSI BASIC 82, p. 115: "A stream file is much like a sequential file, except that it is a sequence of individual values, rather than of records."

118 ANSI BASIC 82, p. 116: "A keyed file as a sequence of records, each of which is identified by a string called a key."

gische Folge der Sätze innerhalb des Files wird durch die Ordnung der Schlüssel bestimmt.[119]
Für die indizierte und relative Fileorganisation hat ANSI BASIC 82 zwei neue Anweisungen in der Form des REWRITE FL-Statements[120] und des DELETE FL-Statements[121] eingeführt.

332.2 Filevereinbarungen

Bei den Filevereinbarungen hat ANSI BASIC 82 das OPEN FL- und CLOSE FL-Statement beibehalten; an die Stelle des SCRATCH FL-Statements[122] tritt das ERASE FL-Statement.[123] Es gilt demnach:

⟨FL-Vereinbarung⟩ : = ⟨OPEN FL-Statement⟩ |
259 258 260

⟨CLOSE FL-Statement⟩ |
271

⟨ERASE FL-Statement⟩
272

332.21 OPEN FL-Statement

Entsprechend den neu genormten Filestrukturen und Fileorganisationen ergeben sich für das OPEN FL-Statement gegenüber ANSI BASIC 79[124] einige Änderungen in den Fileattributen.[125] Zusammengefasst bestehen nunmehr folgende Standards:[126]

⟨Fileattribut⟩ : = ⟨Filetyp⟩ | ⟨Fileorganisation⟩ |
264 263 266 267

⟨Satzgrösse⟩ | ⟨Arbeitsmodus⟩ | §§§
268

119 ANSI BASIC 82, p. 116: "The logical sequence of records is established by the collating order of their keys."
120 Vgl. Abs. 332.41.
121 Vgl. Abs. 332.45.
122 Vgl. Bd. 2, Abs. 332.2 und Abs. 332.23.
123 Vgl. Abs. 332.32.
124 Vgl. Bd. 2, Abs. 332.21.
125 Vgl. Bd. 2, Gleichungen 264–268.
126 ANSI BASIC 82, p. 111–118.

⟨Filetyp⟩ : = RECTYPE ⟨DISPLAY | INTERNAL | NATIVE | ⟨Alphanumerischer Ausdruck⟩⟩
266 264 51

⟨Satzgrösse⟩ : = RECSIZE ⟨VARIABLE | ⟨Alphanumerischer Ausdruck⟩⟩ |
51

RECSIZE ⟨LENGTH | ⟨Numerischer Ausdruck⟩⟩
49

⟨Fileorganisation⟩ : = ORGANIZATION ⟨SEQUENTIAL | STREAM
267 264

RELATIVE | KEYED | ⟨Alphanumerischer Ausdruck⟩⟩
51

⟨Arbeitsmodus⟩ : = ACCESS ⟨INPUT | OUTPUT | OUTIN | ⟨Alphanumerischer Ausdruck⟩⟩
268 264 51

Die Pointerausgangsposition entfällt – im Gegensatz zu ANSI BASIC 79 – als eigenständiges Fileattribut; sie kann in Zukunft in der Adresse[127] angegeben werden.
Die neuen Standards unterscheiden sich demnach hauptsächlich dadurch von ANSI BASIC 79, dass alle Fileattribute durch ein Kennwort (RECTYPE, RECSIZE, ORGANIZATION, ACCESS) eingeleitet werden müssen.
Ausserdem ist der Filename nach ANSI BASIC 82 wie folgt zu spezifizieren:[128]

⟨Filename⟩ : = NAME ⟨Alphanumerischer Ausdruck⟩
262 260 50

127 Vgl. Abs. 332.31.
128 ANSI BASIC 82, p. 111–112.

Korrekte OPEN-Anweisungen sind nach ANSI BASIC 82

```
120 OPEN #N: NAME "FILENAME", ACCESS A$, ORGANIZATION KEYED
200 OPEN #7: NAME B$, ACCESS C$, RECTYP "DISPLAY", ORGANIZATION "sequential"
```

332.22 CLOSE FL-Statement

Für das CLOSE FL-Statement ist die ANSI Norm von 1979 beibehalten worden.[129]

332.23 ERASE FL-Statement

ANSI BASIC 82 ersetzt das Kennwort SCRATCH durch ERASE, wodurch das SCRATCH FL-Statement[130] entsprechend umgenannt wird. Das ERASE FL-Statement weist ausserdem gegenüber dem SCRATCH FL-Statement geringfügige Erweiterungen auf. Durch die – unmittelbar auf das Kennwort ERASE folgende – Angabe REST kann der Filerestinalt gelöscht werden, der nach der aktuellen Pointerposition nach voranden ist.[131] Für das ERASE FL-Statement gilt mithin:[132]

⟨ERASE FL-Statement⟩ : =
272 259

ERASE # ⟨Kanalnummer⟩ | 261
ERASE REST # ⟨Kanalnummer⟩   261

332.3 Fileausgabeanweisungen

Die Fileausgabeanweisungen zeichnen sich nach ANSI BASIC 82 durch praxisorientierte Kennworte aus; anstelle von OUTPUT[133] treten die Kennworte PRINT und WRITE.

129 Vgl. Bd. 2, Abs. 332.22 und ANSI BASIC 82, p. 118.
130 Vgl. Bd. 2, Abs. 332.23.
131 ANSI BASIC 82, p. 119: "If the REST option is specified, then all data beyond the current location of the file pointer are deleted. All data preceding it are left unchanged. The file pointer then points to the new end of file."
132 ANSI BASIC 82, p. 111, 118–119.
133 Vgl. Bd. 2, Abs. 332.3 und ANSI BASIC 79, p. 101.

Die Fileausgabeanweisungen umfassen jene Statements, mit denen datenleere Filebereiche mit Zeichen, Inhalten oder Feldern beschrieben werden können. Es handelt sich daher im eigentlichen Sinne um FL-Erstellungsanweisungen, die von den FL-Modifikationsanweisungen zu unterscheiden sind.[134]
Die Fileausgabeanweisungen umfassen die bereits nach ANSI BASIC 79 vorgesehenen Statementtypen.[135] Im Zusammenhang mit zeichenstrukturierten Files, die durch Fileausgabeanweisungen mit dem Kennwort PRINT aufgebaut werden, sind auch das MARGIN FL-Statement und das ZONEWIDTH FL-Statement eingeführt worden.
Für die Fileausgabeanweisungen gilt mithin:[136]

⟨FL-Ausgabeanweisung⟩ : = ⟨Output FL-Statement⟩ |
273 258 274

⟨Output USING FL-Statement⟩ |
280

⟨MAT Output FL-Statement⟩ |
281

⟨MAT Output USING FL-Statement⟩ |
282

⟨MARGIN FL-Statement⟩ |

⟨ZONEWIDTH FL-Statement⟩

332.31 Output FL-Statement

Für die Datenübertragung aus dem Zentralspeicher auf periphere Geräte dient das Output FL-Statement. Es kann nach ANSI BASIC 82 mit dem Kennwort PRINT oder WRITE beginnen.[137]

134 Vgl. Abs. 332.4.
135 Vgl. Bd. 2, Abs. 332.3, Gleichung 273.
136 ANSI BASIC 82, p. 126–127.
137 ANSI BASIC 82, p. 126.

⟨Output FL-Statement⟩ : =
274 273

PRINT	#	⟨Adresse⟩ 275	:	⟨Outputliste⟩ 279	|
WRITE	#	⟨Adresse⟩ 275	:	⟨Outputliste⟩ 279	

Das PRINT FL-Statement entspricht bis auf die Adresse dem normalen PRINT-Statement;[138] falls die in der Adresse enthaltene Kanalnummer Null ist, besteht bezüglich der Wirkungsweise beider Statements vollständige Korrespondenz.[139] Zu beachten ist, dass mit PRINT FL-Statements – unabhängig von der gewählten Kanalnummer – generell nur zeichenstrukturierte Files aufgebaut werden können.[140]
Das WRITE FL-Statement kann zum Aufbau beliebig strukturierter Files verwendet werden;[141] soweit es sich auf zeichenstrukturierte Files bezieht, entspricht es einem PRINT FL-Statement.[142]
Vermerkt sei, dass die von ANSI BASIC 82 vorgenommene Differenzierung zwischen PRINT FL- und WRITE FL-Statements nicht unbedingt erforderlich ist, da durch das OPEN FL-Statement alle Fileattribute eindeutig festgelegt werden und damit jederzeit über die Kanalnummer überprüfbar sind. Es ist für die zukünftige Normierung zu erwarten, dass die momentan etablierte Bindung zwischen Fileaufbau und Statementkennwort rückgängig gemacht und – im Sinne von ANSI BASIC 79 – ein einheitliches Kennwort für alle Fileausgabeanweisungen festgelegt wird.
Für die Adresse werden nach ANSI BASIC 82 folgende Strukturen vorgesehen:[143]

138 ANSI BASIC 82, p. 128.
139 ANSI BASIC 82, p. 130: "PRINT to channel zero works in accordance with the semantics for a device without record-setter or erase capacity. The destination of the output data is the same as if the channel-expression had been omitted."
140 ANSI BASIC 82, p. 128: "The sequence of characters generated constitutes a record of a DISPLAY file."
141 ANSI BASIC 82, p. 130: "The write- and array-write-statements are used to create records of any type."
142 ANSI BASIC 82, p. 131. "A WRITE operation on a display record generates exactly the same sequence of records and characters as would a PRINT operation with the same expression-list or array-list."
143 ANSI BASIC 82, p. 122–127.

⟨Adresse⟩ : = ⟨Kanalnummer⟩ | ⟨Kanalnummer⟩ , ⟨Pointer-position⟩ |
275 274 261 261 267a

⟨Kanalnummer⟩ , ⟨Pointer-position⟩ , ⟨Ausführungs-überwachung⟩
261 276

⟨Pointerposition⟩ : = BEGIN | NEXT | SAME | END |
276 275

RECORD ⟨Arithmetischer Ausdruck⟩ |
49

KEY ⟨Alphanumerischer Ausdruck⟩ |
51

KEY ⟨Such-operator⟩ | ⟨Alphanumerischer Ausdruck⟩
51

⟨Ausführungs-überwachung⟩ : = ⟨Output-überwachung⟩ | ⟨Modifikations-überwachung⟩ |

⟨Input-überwachung⟩ | ⟨Kontroll-überwachung⟩

Die Ausführungsüberwachung (io-recovery) kann sich mithin auf verschiedene FL-Anweisungstypen beziehen. Zur Überwachung des Programmablaufs in Verbindung mit FL-Ausgabeanweisungen dienen folgende Adresszusätze:

⟨Output-überwachung⟩ : = IF THERE EXIT DO | IF THERE EXIT FOR |

IF THERE ⟨Statementnummer⟩
70

Durch die Output-Überwachung können auch die Aufgaben des nach ANSI BASIC 79 vorgesehenen IF EOF FL-Statements übernommen werden.[144]

144 Vgl. Bd. 2, Abs. 332.53.

Zum Verständnis der Output-Überwachung ist zu beachten, dass FL-Ausgabe-Anweisungen nur auf leere Datenbereiche ausgeführt werden können. Ein in der Output-Überwachung angegebener unbedingter Sprungbefehl wird beim WRITE FL- und PRINT FL-Statement ausgeführt, wenn der angegebene Schlüssel in einem indizierten File vorhanden ist, der Pointer nicht am Fileende steht und in der aktuellen Position auf einen bereits beschriebenen Satz deutet.[145] Diesfalls wird die Datenübertragung aus der Outputliste nicht durchgeführt, sondern mit dem Output-Überwachungsbefehl fortgefahren; es kommt zu keinem Programmabbruch. Falls dagegen eine Angabe zur Output-Überwachung (io-recovery) fehlt, erfolgt eine Fehlermeldung, sofern das Statement nicht in einem Ausnahmeblock liegt.[146]
Bezüglich der Angabe zur Pointerposition ist zu beachten, dass BEGIN, NEXT, SAME und END auf alle Fileorganisationsformen angewendet werden können. Die Angabe RECORD mit nachfolgendem arithmetischen Ausdruck kann nur bei relativ organisierten Files Verwendung finden. Durch die Angabe KEY als Pointerposition wird automatisch ein indiziertes File angesprochen und der Suchbegriff auf Vorhandensein geprüft.[147] Wird eine Pointerpositionsangabe auf eine nicht zuständige Fileorganisationsform – beispielsweise die KEY-Angabe auf ein sequentielles File – ausgeführt, so erfolgt eine Fehlermeldung, falls das Statement nicht in einem Ausnahmeblock[148] enthalten ist.
Im allgemeinen wird das Kennwort KEY ohne explizite Angabe eines Suchoperators verwendet; bei FL-Ausgabeanweisungen gilt im übrigen nur das Gleichheitszeichen.

⟨Suchoperator⟩ : = =

Durch das Statement

```
200 PRINT #3, END: A, B; C, D
```

werden über den Kanal mit der Nummer 3 an das Fileende in einem Satz die Inhalte von A, B, C und D angefügt, wobei B auf A im Zonenabstand, dann C im gepackten Abstand[149] und D wiederum im Zonenabstand aufeinander folgen.

145 ANSI BASIC 82, p. 124.
146 Zum Ausnahmeblock vgl. Abs. 329.2.
147 ANSI BASIC 82, p. 122–124.
148 Vgl. Abs. 329.2.
149 Zur gepackten Schreibweise vgl. Bd. 1, S. 175–179.

```
350 WRITE #N, RECORD 47, IF THERE 678: A+B, C$ & D$
```

Mit dieser Anweisung wird auf dem Kanal N der siebenundvierzigste Satz, wenn er noch leer ist, mit den Inhalten der beiden Ausdrücke der Outputliste gefüllt. Dies unter der Voraussetzung, dass mit dem Kanal N ein inhaltsstrukturiertes File angesprochen wird.
Zusammenfassend ist festzuhalten, dass bei einem PRINT FL- oder WRITE FL-Statement drei Phasen der Statementausführung zu unterscheiden sind. Die erste Phase überprüft aufgrund der angegebenen Kanalnummer, ob die geplante Operation mit den im OPEN FL-Statement[150] spezifizierten Fileattributen vereinbar sind. Ist dies der Fall, so wird die Ausführbarkeit durch die Output-Ausführungsüberwachung geprüft und erst dann in der dritten Phase die eigentliche Datenübertragung vorgenommen.[151]

332.32 Output USING FL-Statement

Das Output USING FL-Statement unterscheidet sich nach ANSI BASIC 82 zunächst durch die Kennworte PRINT und WRITE von ANSI BASIC 79. Weiterhin wird die Formatierung nur im PRINT USING FL-Statement durch das Kennwort USING, ansonsten mit WITH, eingeleitet.[152] Die Formatierung wird an die Adresse angehängt und erscheint damit vor der Outputliste.[153]

⟨Output USING FL-Statement⟩ : =
280 273

PRINT	#	⟨Adresse⟩ 275	,	⟨Imagekennung⟩	:	⟨Outputliste⟩ 279	|
WRITE	#	⟨Adresse⟩ 275	,	⟨Templatekennung⟩	:	⟨Outputliste⟩ 279	

Die Angaben zur Formatierung erfolgen in der Image- resp. Templatekennung.

150 Vgl. Bd. 2, Abs. 332.21 und Abs. 332.21.
151 ANSI BASIC 82, p. 128.
152 ANSI BASIC 82, p. 126–127.
153 Vgl. Bd. 2, Abs. 332.32.

⟨Imagekennung⟩ : = USING ⟨Image-Statementkennung⟩
131

⟨Templatekennung⟩ : = WITH ⟨Image-Statementkennung⟩
131

Aus den obigen Strukturgleichungen ist ersichtlich, dass ausser der Adressänderung nur im WRITE WITH FL-Statement im Vergleich zu ANSI BASIC 79 eine andere Form des Formatierungsaufrufs erfolgt. Als Template kann entweder die Statementnummer des entsprechenden TEMPLATE-Statements[154] oder direkt die Schablonenliste des TEMPLATE-Statements als alphanumerischer Ausdruck angegeben werden.[155]

332.33 MAT Output FL-Statement

eim MAT Output-Statement betreffen die Neuerungen gemäss ANSI ASIC 82 nur die Adressangaben, so dass weiterhin gilt:[156]

⟨MAT Output FL-Statement⟩ : =
281 273

MAT PRINT # ⟨Adresse⟩ : ⟨DIM-gebundene Ausgabeliste⟩ |
275 254
MAT WRITE # ⟨Adresse⟩ : ⟨DIM-gebundene Ausgabeliste⟩
275 254

332.34 MAT Output USING FL-Statement

Beim MAT Output USING FL-Statement betreffen die Neuerungen nach ANSI BASIC 82 ebenfalls die Adressangabe und, beim MAT WRITE WITH FL-Statement, die andersartige Form des Formataufrufs.[157]

154 Zum TEMPLATE-Statement vgl. Abs. 332.11.
155 Die Templatekennung ist damit bis auf das Kennwort identisch mit der Image-Statementkennung; vgl. Bd. 1, S. 186.
156 ANSI BASIC 82, p. 126.
157 ANSI BASIC 82, p. 126.

⟨MAT Output USING FL-Statement⟩ : =
282 273

MAT PRINT # ⟨Adresse⟩ , ⟨Imagekennung⟩ : ⟨DIM-gebundene MAT-Ausgabeliste⟩
275 254

MAT WRITE # ⟨Adresse⟩ , ⟨Template-kennung⟩ : ⟨DIM-gebundene MAT-Ausgabeliste⟩
275 254

332.35 MARGIN FL-Statement

Das MARGIN FL-Statement ist die fileorientierte, durch die Kanalnummer ergänzte, Anpassung des einfachen MARGIN-Statements.[158]

⟨MARGIN FL-Statement⟩ : =

MARGIN # ⟨Kanalnummer⟩ : ⟨Arithmetischer Ausdruck⟩
261 49

Die Bedingungen und die Wirkungsweise entsprechen vollständig dem einfachen MARGIN-Statement,[159] wobei lediglich die Ausführung auf einem peripheren Gerät erfolgt.

332.36 ZONEWIDTH FL-Statement

Durch das ZONEWIDTH FL-Statement kann die Zonenzeichenzahl des Standardoutputs von der Normalzahl 15 abgeändert werden. Die Wirkungsweise ist analog dem normalen ZONEWIDTH-Statement[160] und bezieht sich speziell auf das über die Kanalnummer angesprochene periphere Gerät. ZONEWIDTH FL- und MARGIN FL-Statement sind nur

158 ANSI BASIC 82, p. 126, 129.
159 Vgl. Abs. 328.
160 Vgl. Abs. 328.

in Verbindung mit PRINT FL- und MAT PRINT FL-Statements einsetzbar und beziehen sich nur auf zeichenstrukturierte Files.[161]

⟨ZONEWIDTH FL-Statement⟩ : =

ZONEWIDTH # ⟨Kanalnummer⟩ : ⟨Arithmetischer Ausdruck⟩
261 49

Durch die Anweisung

```
100 MARGIN # 7: 120
```

wird die Zeichenzahl für Sätze aus dem über Kanalnummer 7 angesprochenen File von 75 auf 120 erhöht und mit

```
110 ZONEWIDTH # 7: 40
```

wird jede Zone in der Zeichenzahl von 15 auf 40 erhöht. Die Zonenzahl reduziert sich von 5 auf 3.
Die Wirkung des MARGIN FL- und ZONEWIDTH FL-Statement tritt ab der aktuellen Pointerposition ein und kann daher auch nur einen Teil eines Satzes betreffen.

332.4 Filemodifikationsanweisungen

Den FL-Ausgabeanweisungen stehen die FL-Modifikationsanweisungen gegenüber. Sie dienen dazu, datenbesetzte Filebereiche in ihrem Dateninhalt zu ändern.
Nach ANSI BASIC 82 sind folgende Statementtypen vorgesehen:[162]

⟨FL-Modifikations-anweisung⟩ : = ⟨REWRITE FL-Statement⟩ |

⟨REWRITE WITH FL-Statement⟩ |

⟨MAT REWRITE FL-Statement⟩ |

161 ANSI BASIC 82, p. 129–130.
162 ANSI BASIC 82, p. 146–149.

⟨MAT REWRITE WITH FL-Statement⟩ |

⟨DELETE FL-Statement⟩

Zu beachten ist, dass die vorerwähnten Statements auf alle Filetypen anwendbar sind. Auch die mit PRINT FL-, PRINT USING FL-, MAT PRINT FL- oder MAT PRINT USING FL-Statements aufgebauten Files können nur mit REWRITE FL-, MAT REWRITE FL- und DELETE FL-Statements geändert werden. Dies kann insbesondere bei Sätzen, die USING-bezogen aufgebaut wurden, zu Komplikationen führen, da in REWRITE FL- und MAT REWRITE FL-Statements keine Imagekennung zulässig ist. Ein mit dem Kennwort PRINT USING-bezogen erstellter Satz muss deshalb zuerst durch ein DELETE FL-Statement gelöscht werden, bevor er mit einem PRINT USING FL-Statement neu aufgebaut werden kann.

332.41 REWRITE FL-Statement

Nach ANSI BASIC 82 entspricht der Aufbau des REWRITE FL-Statements bis auf das Kennwort und eine eventuell in der Adresse enthaltene Modifikationsüberwachung der Grundstruktur des WRITE FL-Statements.[163]

⟨REWRITE FL-Statement⟩ :=

REWRITE	# ⟨Adresse⟩ 275	:	⟨Outputliste⟩ 279

Für die eventuell in der Adresse enthaltene Modifikationsüberwachung gilt:[164]

⟨Modifikations-überwachung⟩ := IF MISSING EXIT DO |
IF MISSSING EXIT FOR |
IF MISSING ⟨Statementnummer⟩ 70

163 ANSI BASIC 82, p. 146–147.
164 ANSI BASIC 82, p. 84, 146; vgl. auch Abs. 332.31.

Durch die Modifikations-Überwachung wird die Existenz eines Satzes geprüft; falls er fehlt, wird der mit einem REWRITE-Statement verbundene Löschvorgang nicht ausgeführt, sondern die zur Überwachung des Programmablaufs angegebene Anweisung ausgeführt und dabei die aktuelle Position des Filepointers beibehalten.[165]
Als Suchoperator ist in einem KEYED organisierten File nur das Gleichheitszeichen erlaubt.

332.42 REWRITE WITH FL-Statement

Beim REWRITE WITH FL-Statement wird der Adresse eine Templatekennung angefügt;[166] der Rest des Statements ist mit dem REWRITE FL-Statement identisch.[167]

⟨REWRITE WITH FL-Statement⟩ :=

REWRITE = ⟨Adresse⟩ 275 , ⟨Templatekennung⟩ : ⟨Outputliste⟩ 279

332.43 MAT REWRITE FL-Statement

Durch das MAT REWRITE FL-Statement können ganze Felder zur Modifikation von Sätzen herangezogen werden. Die Struktur des Statements ist bis auf das Kennwort und eine eventuell in der Adresse enthaltene Modifikationsüberwachung[168] mit dem MAT WRITE FL-Statement identisch.[169]

⟨MAT REWRITE FL-Statement⟩ : =

MAT REWRITE # ⟨Adresse⟩ 275 : ⟨DIM-gebundene Ausgabeliste⟩ 254

165 ANSI BASIC 82, p. 147: "If there is an exception, the pointer is left as it was set ... and the data in the file is unchanged."
166 Zur Templatekennung, die auch das Kennwort WITH umfasst, vgl. 332.32.
167 ANSI BASIC 82, p. 146; vgl. auch Abs. 332.41.
168 Vgl. Abs. 332.31.
169 ANSI BASIC 82, p. 146.

332.44 MAT REWRITE WITH FL-Statement

Ebenso wie das REWRITE FL-Statement kann auch das MAT REWRITE FL-Statement durch eine Templatekennung nach der Adresse ergänzt werden, wodurch jede beliebige Form des Satzaufbaus möglich wird.[170]

⟨MAT REWRITE WITH FL-Statement⟩ : =

MAT REWRITE # ⟨Adresse⟩ 275 , ⟨Template-kennung⟩ : ⟨DIM-gebundene Ausgabeliste⟩ 254

332.45 DELETE FL-Statement

Das DELETE FL-Statement[171] dient zum Inhaltslöschen einzelner datenbesetzter Sätze eines Files.[172] Seine Struktur ist einfach; es besteht nur aus Kennwort und Adresse.

⟨DELETE FL-Statement⟩ : =

DELETE # ⟨Adresse⟩ 275

Die Adresse kann wiederum eine Modifikationsüberwachung enthalten, die bei Nichtexistenz des zu löschenden Satzes ausgeführt wird.[173]
Nach ANSI BASIC 82 korrekte Filemodifikationsanweisungen sind:

```
100 REWRITE # 7, KEY = B$: B$, C$
250 MAT REWRITE # N, RECORD L-1, IF MISSING 2333: T$(3:45)
700 REWRITE # N, KEY "JONES", WITH 2250: L1, L2, N, K$
800 DELETE # 6,   RECORD 22, IF MISSING EXIT FOR
900 REWRITE # 6, SAME, WITH 222: A, B, C, A1
950 DELETE #8
```

170 ANSI BASIC 82, p. 146.
171 ANSI BASIC 82, p. 147–148.
172 Vgl. dazu auch das ERASE FL-Statement in Abs. 332.23, das zum Inhaltslöschen ganzer Filebereiche dient. Von der Wirkungsweise her könnte das ERASE FL-Statement auch den FL-Modifikationsanweisungen zugeordnet werden.
173 Vgl. Abs. 332.31.

Nach ANSI BASIC 82 sollen die verschiedenen Formen des REWRITE FL-Statements nur auf relative oder indizierte Files angewendet werden.[174] Sequentielle oder fortlaufend organisierte Files können demnach nur durch eine Kombination von DELETE FL- und PRINT FL- bzw. WRITE FL-Statements in ihren Datenbeständen modifiziert werden. Vgl. hierzu das nachstehende Beispiel.

```
200 DELETE # 3
210 PRINT # 8, SAME: A$, N1, N2
```

Ebenso lassen sich alle Formen des REWRITE FL-Statements durch eine Kombination von vorstehenden DELETE FL- mit logisch nachfolgenden WRITE FL-Statements ersetzen.

```
400 DELETE # N, KEY "JONES"
410 MAT REWRITE # N, KEY "JONES": P$, P1, P2, P3
```

332.5 *Fileeingabeanweisungen*

Nach ANSI BASIC 82 wird die Zahl der verfügbaren Fileeingabeanweisungen im Vergleich zu ANSI BASIC 79 wesentlich erhöht. Zusätzlich zum Kennwort INPUT kann – als Korrelat zu den in den Fileausgabeanweisungen ebenfalls zulässigen WRITE – das Kennwort READ benutzt werden.[175]
Während die INPUT FL-Statements nur auf zeichenstrukturierte, sequentiell organisierte Files angewendet werden dürfen, sind für die READ FL-Statements alle Satztypen und Organisationsformen zugelassen.[176]

⟨FL-Eingabeanweisung⟩ : = ⟨Input FL-Statement⟩ | ⟨READ WITH FL-Statement⟩ | ⟨MAT Input FL-Statement⟩ | ⟨MAT READ WITH FL-Statement⟩

283 258 284 268

174 ANSI BASIC 82, p. 108.
175 ANSI BASIC 82, p. 136–145.
176 ANSI BASIC 82, p. 108.

Damit besteht hinsichtlich der Grundeinteilung Übereinstimmung mit den FL-Erstellungsanweisungen,[177] wobei im Sinne von ANSI BASIC 82 das Kennwort WITH (anstelle von USING) verwendet wird.

332.51 Input FL-Statements

Nach ANSI BASIC 82 können Input FL-Statements in fünf verschiedenen Formen realisiert werden:

⟨Input FL-Statement⟩ : =
284 283

```
INPUT # ⟨Adresse⟩ : ⟨Ergänzte Inputliste⟩ |
          275

INPUT # ⟨Adresse⟩ , ⟨Identifikation⟩ : ⟨Ergänzte Inputliste⟩ |
          275

LINE INPUT # ⟨Adresse⟩ : ⟨Alphanumerische Variablenliste⟩ |
               275                                  117d

LINE INPUT # ⟨Adresse⟩ , ⟨Identifikation⟩ : ⟨Alphanumerische⟩ |
               275                          Variablenliste
                                                    117d
READ # ⟨Adresse⟩ : ⟨Ergänzte Inputliste⟩
          275
```

Gegenüber ANSI BASIC 79 weist das INPUT FL-Statement zwei Modifikationen auf.[178] Verwiesen sei auf die neue Adressform, die – dem normalen INPUT-Statement entsprechende – Identifikation und die ergänzte Inputliste.[179]
Die Adresse weist eine dem Output FL-Statement entsprechende Struktur auf. Die Angabe der Pointerposition ist allerdings auf die Kennworte BEGIN, SAME, NEXT und END begrenzt.[180] Daraus ergibt sich, dass

177 Vgl. Abs. 332.3.
178 ANSI BASIC 82, p. 136.
179 ANSI BASIC 82, p. 136–139.
180 Zur Pointerposition im allgemeinen vgl. 332.31.

das INPUT FL-Statement nur auf sequentielle Files und dort nur auf zeichenstrukturierte Sätze anwendbar ist.[181]
Für die eventuell in der Adresse enthaltene Input-Überwachung gilt (analog zur Modifikationsüberwachung):

⟨Input-Überwachung⟩ : = IF MISSING EXIT DO | IF MISSING EXIT
IF MISSING ⟨Statementnummer⟩
70

Die Inputliste kann um den Zusatz SKIP REST ergänzt werden, wodurch der Pointer bei teilweisem Lesen eines Satzes über den Satzrest auf den Anfang des nächsten Satzes gesetzt wird.[182] Ohne den Zusatz verbleibt der Pointer in der Position, die er nach Ausführung des INPUT FL-Statements erreicht hat; ein Komma am Ende der Inputliste ist – im Gegensatz zu ANSI BASIC 79 – nicht mehr zulässig.[183]

⟨Ergänzte Inputliste⟩ : = ⟨Inputliste⟩ | ⟨Inputliste⟩ , SKIP REST
285 285

Das LINE INPUT FL-Statement ist die fileorientierte Form des LINE INPUT-Statements.[184] ANSI BASIC 82 sieht allerdings die beim INPUT FL-Statement mögliche Ergänzung der alphanumerischen Variablenliste[185] durch SKIP REST hier nicht vor.[186]
Normgerechte Beispiele für Fileeingabeanweisungen mit den Kennworten INPUT oder LINE INPUT stellen die nachfolgenden Statements dar:

```
200  INPUT # N, BEGIN, IF MISSING EXIT DO: A, B, C
210  INPUT #3, NEXT, TIMEOUT 20: I$, J$
220  LINE INPUT # R1, SAME, IF MISSING 666: A$(7,8)
230  READ # 7, RECORD R,: A, B, SKIP REST
```

181 ANSI BASIC 82, p. 139: "All the INPUT operations (as opposed to the READ operations) are valid only for a file opened as DISPLAY."
182 ANSI BASIC 82, p. 138: "The SKIP REST option is allowed only for non-stream files. It causes the remainder of the record from which the last datum or value or field was taken to be ignored."
183 Vgl. Bd. 2, Abs. 332.41.
184 Vgl. Bd. 1, Abs. 332.14 und S. 169.
185 Mit den verschiedenen Formen des LINE INPUT- bzw. LINE INPUT FL-Statements können nur Daten in alphanumerische Variable eingelesen werden.
186 ANSI BASIC 82, p. 136.

Aus dem syntaktischen Aufbau dieser Fileanweisungen ergibt sich, dass als periphere Geräte und damit als Träger der physischen Dateien nicht nur Magnetplatten oder Magnetbänder in Betracht kommen, sondern alle Arten von Dateneingabegeräten. Als solche kommen insbesondere auch Bildschirmterminals in Frage, für die die Identifikationsangaben relevant sind.[187] Bei physischen Dateien bleiben eventuelle Identifikationsangaben wirkungslos; wird als Kanalnummer eine Null angegeben, so entsteht automatisch die Wirkungsweise des normalen INPUT-Statements.[188]
Das READ-Statement erlaubt die Verarbeitung aller Satz- und Organisationsarten von Files.[189]
Die in der Adresse mitenthaltene Pointerpositionsangabe weitet sich im Vergleich zum INPUT FL-Statement – bei dem nur die Kennworte BEGIN, NEXT, SAME und END zulässig sind – auf den für das Output FL-Statement festgelegten Umfang aus.[190]
Die ergänzte Inputliste ist identisch mit der des INPUT FL-Statements.
Die Angabe SKIP REST erlaubt es, den nicht verarbeiteten Rest eines Satzes zu überspringen, so dass der Pointer nach Ausführung eines derartigen READ-Statements auf dem Anfang des nächsten Satzes steht.[191]
Als Suchoperator ist bei indizierter Fileorganisation nur das Gleichheitszeichen zugelassen.

332.52 READ WITH FL-Statement

Das READ WITH FL-Statement wird verwendet, um feldstrukturierte Sätze,[192] die mit WRITE WITH FL- oder MAT WRITE WITH FL-Statements aufgebaut wurden,[193] einzulesen.

⟨READ WITH FL-Statement⟩ :=

READ # ⟨Adresse⟩ , ⟨Templatekennung⟩ : ⟨Ergänzte Inputliste⟩

187 ANSI BASIC 82, p. 138: "Input-control-items [such as a prompt-specifier, timeout-expression, and time-inquiry] apply only to interactive terminal devices."
188 ANSI BASIC 82, p. 140: "Input from channel zero works in accordance with the semantics for non-file devices."
189 ANSI BASIC 82, p. 108.
190 Vgl. Abs. 332.311; zulässig sind die Kennworte BEGIN, NEXT, SAME, END, RECORD und KEY.
191 ANSI BASIC 82, p. 138.
192 Vgl. Abs. 332.11.
193 Vgl. Abs. 332.31, Abs. 332.32.

Vermerkt sei, dass von ANSI BASIC 82 nur das READ WITH FL-Statement strukturmässig explizit dargestellt wird; ein READ USING FL-Statement erscheint im Rahmen eines Beispiels.[194]

```
READ #3, SAME, USING 333: W$, SKIP REST
```

Vorschlagsgemäss sollte es wie folgt lauten:

```
READ #3, SAME, WITH 333: W$, SKIP REST
```

332.53 MAT Input FL-Statement

Nach ANSI BASIC 82 sind fünf Statementformen vorgesehen.[195] Sie sind bis auf das Kennwort und den entsprechenden feldorientierten MAT-Input- bzw. MAT-Readlisten identisch mit den skalaren Input FL-Statements.[196]

⟨MAT Input FL-Statement⟩ : =
286 283

MAT INPUT # ⟨Adresse⟩ : ⟨MAT-Inputliste⟩ |
275 227a

MAT INPUT # ⟨Adresse⟩ , ⟨Identifikation⟩ : ⟨MAT-Inputliste⟩ |
275 227a

MAT LINE INPUT # ⟨Adresse⟩ : ⟨MAT-Inputliste⟩ |
275 227a

MAT LINE INPUT # ⟨Adresse⟩ , ⟨Identifikation⟩ : ⟨MAT-Inputliste⟩ |
275 227a

MAT READ # ⟨Adresse⟩ : ⟨MAT-Readliste⟩
275

194 ANSI BASIC 82, p. 136–137.
195 ANSI BASIC 82, p. 136–137, 139–140, 142–143.
196 Vgl. Abs. 332.51.

Zu beachten ist, dass beim MAT INPUT FL-Statement über die MAT-Inputliste auch ein variabler Vektor festgelegt werden kann;[197] die – in Verbindung mit dem MAT READ FL-Statement zu verwendende – MAT-Readliste umfasst dagegen nur die DIM-gebundene und DIM-begrenzte Eingabeliste.[198]

⟨MAT-Readliste⟩ : = ⟨DIM-gebundene MAT-Eingabeliste⟩ (214) | ⟨DIM-begrenzte MAT-Eingabeliste⟩ (216) |

⟨MAT-Readliste⟩ , ⟨MAT-Readliste⟩

Vgl. hierzu die folgenden Beispiele:

```
210 MAT INPUT #5, NEXT, TIMEOUT 30: K$, L(8), V(7)
300 MAT LINE INPUT #3, SAME, PROMPT "81 TEXTZEILEN EINGEBEN": B$(3,9)
500 MAT READ #7, RECORD N, WITH 888, IF MISSING 999: L
```

Das MAT READ FL-Statement ermöglicht die ganzheitliche Datenfeldeingabe von peripheren Datenträgern. Semantisch entspricht es vollständig dem normalen MAT READ-Statement.[199] Durch die Adressangabe wird die entsprechende fileorientierte Verarbeitungsart bewirkt, indem vor der eigentlichen Datenübertragung in die MAT-Readliste eine kontrollierte Pointerversetzung mit eventueller Ausführungsüberwachung erfolgen kann.

332.54 MAT READ WITH FL-Statement

Nach ANSI BASIC 82 ist die folgende Form vorgesehen:[200]

⟨MAT READ WITH FL-Statement⟩ : =

MAT READ # ⟨Adresse⟩ , ⟨Templatekennung⟩ (275) : ⟨MAT-Readliste⟩

197 Vgl. Bd. 2, Abs. 331.12.
198 ANSI BASIC 82, p. 136.
199 ANSI BASIC 82, p. 142: "In general, an array-read-statement behaves just like the read-statement would if the arrays were written out explicitly as array-elements."
200 ANSI BASIC 82, p. 136–137.

Durch das Statement

```
300 MAT READ # 7, RECORD 48, WITH 650, IF MISSING 600: A, B(A(2))
```

wird in einem inhaltstrukturierten File von Satz 48 – nach der in Statement 650 angegebenen Maske – zunächst numerische Werte in A und dann weitere Werte bis zum entsprechenden Element des Vektors B eingelesen. Falls der Satz 48 leer ist, wird das Programm mit Statementnummer 600 fortgesetzt. Die Wertinhalte von A und B bleiben dann unbeeinflusst.

332.6 Filekontrollanweisungen

Bei den Filekontrollanweisungen entfällt nach ANSI BASIC 82 das IF EOF FL-Statement,[201] da seine Wirkungsweise in den Ausnahmeblock und die Ausführungskontrolle[202] übertragen wurde.
Das Inquire FL-Statement[203] ist um einige Attribute ergänzt und durch ein neues Kennwort ASK in ein ASK FL-Statement umbenannt.
Das Restore FL-Statement[204] wird durch ein neues Kennwort RESET nunmehr als RESET FL-Statement bezeichnet.
Für die Filekontrollanweisungen gilt mit ANSI BASIC 82:

⟨FL-Kontrollanweisung⟩ : = ⟨Restore FL-Statement⟩ |
288 258 289

⟨Inquire FL-Statement⟩
290

332.61 Restore FL-Statement

Das Restore FL-Statement dient zur Manipulation des Filepointers. Nach ANSI BASIC 82 ist folgende Statementstruktur vorgesehen:[205]

201 Vgl. Bd. 2, Abs. 332.53 und Bd. 3, Abs. 332.31.
202 Vgl. Abs. 329.3.
203 Vgl. Bd. 2, Abs. 332.52.
204 Vgl. Bd. 2, Abs. 332.51.
205 ANSI BASIC 82, p. 122.

⟨Restore FL-Statement⟩ : =
289 288

RESET # ⟨Adresse⟩
275

Im Vergleich zu ANSI BASIC 79 wird das Kennwort RESET anstelle von RESTORE benutzt;[206] die Adresse kann – neben den zur Pointerpositionierung erforderlichen Angaben – auch eine Kontroll-Überwachung enthalten, wobei

⟨Kontroll-Überwachung⟩ : = ⟨Output-Überwachung⟩ |
⟨Input-Überwachung⟩

gilt.[207]
Zur Pointerpositionierung mit KEY ist die explizite Angabe eines Suchoperators dann erforderlich, wenn nicht das Gleichheitszeichen benutzt wird. Im Gegensatz zu den übrigen FL-Anweisungen gilt:[208]

⟨Suchoperator⟩ : = = | > | > =

Der Suchoperator bestimmt den Charakter des Suchprozesses. Falls das Gleichheitszeichen als Suchoperator bzw. nur das Kennwort KEY verwendet wird, liegt exaktes Suchen (exact-search) vor, andernfalls wird von unexaktem Suchen (inexact-search) gesprochen.[209]

Bei exaktem Suchen wird der Pointer auf den angegebenen Satz versetzt; existiert der angegebene Schlüssel nicht und fehlt eine Kontroll-Überwachung, so wird der Pointer zum nächstgrösseren Schlüssel oder – falls ein solcher nicht vorhanden ist – zum Fileende versetzt.[210]
Bei der unexakten Suchart kann der Pointer auf einen Satz gesetzt werden, dessen Schlüssel im Ordnungssinne grösser oder grösser/gleich dem

206 Vgl. Bd. 2, Abs. 332.51.
207 Vgl. Abs. 332.31, Abs. 332.41.
208 ANSI BASIC 82, p. 122.
209 ANSI BASIC 82, p. 124.
210 ANSI BASIC 82, p. 124: "For an exact-search the pointer is set to the record whose key equals that of the string-expression; if none such exists, the pointer is set to the first record whose key is greater than the string-expression. If there is no such record, the pointer is set to the end of file."

angegebenen Schlüssel ist; wenn kein grösserer Schlüssel existiert, wird der Pointer zum Fileende versetzt.[211]
Im Vergleich zu ANSI BASIC 79 ist zu beachten, dass im Restore FL-Statement eine Pointerpositionsangabe unbedingt enthalten sein muss.[212] Ausserdem darf ein Restore FL-Statement nur auf Files ausgeführt werden, deren physikalische Realisation einen derartigen Vorgang zulässt.[213] Die logische Kombination von RESET FL-Statement und DELETE FL-Statement kann alle Formen des REWRITE FL-Statements ersetzen. Das nachstehende Beispiel zeigt dies für ein relativ organisiertes File.

```
100 RESET # N, RECORD I, IF MISSING 120
110 DELETE # N, SAME
120 WRITE # N, SAME: A, B$, C
  ....
  ....
```

332.62 Inquire FL-Statement

Beim Inquire FL-Statement muss nach ANSI BASIC 82 das Kennwort ASK verwendet werden; im übrigen bleibt die nach ANSI BASIC 79 vorgesehene Statementstruktur erhalten.[214]

⟨Inquire FL-Statement⟩ : =
290 288

ASK # ⟨Kanalnummer⟩ : ⟨Inquireliste⟩
261 291

In der Inquireliste treten einige Ergänzungen auf; Einzelheiten sind aus Tabelle 2 ersichtlich.[215]

211 ANSI BASIC 82, p. 124: "For an inexact-search with not-less, the pointer is set exactly as for an exact-search. For an inexact-search with greater-than-sign, the pointer is set to the first record whose key is strictly greater than the string-expression; if none exists, it is set to the end of file".
212 Vgl. Bd. 2, Abs. 332.51; bei fehlender Pointerpositionsangabe wird nach ANSI BASIC 79 auf den Fileanfang positioniert.
213 ANSI BASIC 82, p. 122–123: "A device may not be able to achieve the effect of a record-setter".
214 ANSI BASIC 82, p. 111–112, 119–120.
215 ANSI BASIC 82, p. 119–120; vgl. auch Bd. 2, Abs. 332.52.

Tabelle 2. Inquireangaben nach ANSI BASIC 82

Attribut	Variable	Übergabewert
ACCESS	alphanumerisch	Arbeitsmodus: INPUT | OUTPUT | OUTIN
CAPABILITY	alphanumerisch	Kanalzuordnung: FILE DEVICE: ab Bei Zuordnung peripherer Geräte wird mit a die Pointerversetzungsmöglichkeit und b die Datenlöschungsmöglichkeit angegeben, wobei a und b die Werte von Y (= ja) oder N (= nein) annehmen können.
COLLATE	alphanumerisch	Schlüsselsortierung, bei indizierten Files: STANDARD | NATIVE
DATUM	alphanumerisch	Nächstfolgender Datentyp ab aktueller Pointerposition: NUMERIC | STRING | NONE | UNKNOWN
KEY	alphanumerisch	Satz identifizierender Schlüssel wird angegeben.
MARGIN	numerisch	Maximale Zeichenzahl je Satz, wobei 0 = beliebig, −1 = kein zeichenstrukturiertes File bedeutet.
NAME	alphanumerisch	Filename
ORGANIZATION	alphanumerisch	Fileorganisationsart: SEQUENTIAL | STREAM | RELATIVE | KEYED
POINTER	alphanumerisch	Aktuelle Pointerposition: BEGIN | MIDDLE | END

Fortsetzung Tabelle 2.

Attribut	Variable	Übergabewert
RECORD	numerisch	Satznummer bei aktueller Pointerposition, 0 bei nicht relativer Fileorganisation.
RECSIZE	alphanumerisch numerisch	VARIABLE | Satzgröße
RECTYPE	alphanumerisch	Satztyp: DISPLAY | INTERNAL | NATIVE
ZONEWIDTH	numerisch	Zeichenzahl je Zone, bei zeichenstrukturierten Files.

333 Erweiterte Programmsteuerungsanweisungen

ANSI BASIC 82 erlaubt die Definition externer Funktionen, die zur Gruppe der Subroutineanweisungen zu zählen sind. Die Programmsegmentierungsanweisungen bleiben in Aufbau und Wirkungsweise unverändert.

331.1 Subroutineanweisungen

Die Subroutineanweisungen sind um einen FUNCTION-Block erweitert.[216] Er stellt – in Analogie zum SUB-Block – ein eigenständiges Unterprogramm dar.
Für die Subroutineanweisungen gilt gemäss ANSI BASIC 82:

⟨Subroutineanweisung⟩ : = ⟨CALL-Statement⟩ | ⟨SUB-Block⟩ |
296 295 297 306

⟨FUNCTION-Block⟩

216 ANSI BASIC 82, p. 72–73.

333.11 CALL-Statement

Das nach ANSI BASIC 82 vorgesehene CALL-Statement weicht in bezug auf Ausbau und Wirkungsweise nicht von ANSI BASIC 79 ab;[217] vgl. Bd. 2, Abs. 333.11.

333.12 SUB-Block

Bezüglich des SUB-Blocks betreffen die Normänderungen von ANSI BASIC 82 nur das SUBEXIT- und SUBEND-Statement; beide weisen neue Kennworte auf, so dass nunmehr gilt:[218]

⟨SUBEXIT-Statement⟩ : =
313 307

EXIT SUB

⟨SUBEND-Statement⟩ : =
314 307

END SUB

333.13 FUNCTION-Block

Der FUNCTION-Block weist hinsichtlich Aufbau und Wirkungsweise grosse Ähnlichkeit mit dem DEF-Block auf.[219]

⟨FUNCTION-Block⟩ : =

⟨FUN-Statement⟩
⟨Interner FUNCTION-Block⟩
⟨FUNEND-Statement⟩

217 Vgl. ANSI BASIC 82, p. 78.
218 ANSI BASIC 82, p. 78.
219 ANSI BASIC 82, p. 72–73; vgl. Abs. 325.2.

Für die einzelnen Blockelemente gelten folgende Definitionsgleichungen:

⟨FUN-Statement⟩ : = FUNCTION ⟨Funktionsindikator⟩

⟨Funktionsindikator⟩ : = ⟨Numerischer Funktions-indikator⟩ | ⟨Alphanumerischer Funktions-indikator⟩

102a 106

⟨Interner FUNCTION-Block⟩ : = ⟨FUNEXIT-Statement⟩ | ⟨Statementblock⟩ |

⟨Statementblock⟩ | ⟨Interner FUNKTION-block⟩ |

⟨FUNEXIT-Statement⟩ : = EXIT FUNCTION

⟨FUNEND-Statement⟩ : = END FUNCTION

Der Unterschied zwischen FUNCTION- und DEF-Block liegt allein in dem lokalen Begrenzen der Variablennamen und Variablenwerte auf den Bereich des FUNCTION-Blocks. Alle im FUNCTION-Block definierten Variablennamen sind vollkommen wertunabhängig von eventuell identischen Variablennamen des aufrufenden Programms. Globale Wertabhängigkeit besteht lediglich bei den Variablen der Parameterliste.[220]
Das folgende Funktionsprogramm übernimmt die an vielen Programmstellen oft notwendige Abfrage auf eine Ja- oder Nein-Entscheidung und übergibt in der Variablen ANTWORT eine 1 für JA und eine 2 für NEIN.

```
600 FUNCTION ANTWORT
610 DO
620     PRINT "JA ODER NEIN ?"
630     INPUT A$
640     SELECT A$
650     CASE "JA", "ja"
660             ANTWORT = 1
```

220 ANSI BASIC 82, p. 76: "The treatment of variables and arrays which are not parameters in a function-definition shall depend upon whether the function-definition is internal or external. If the function-definition is external, then such variables and arrays shall be local to each invocation of that program-unit, i. e., they shall be distinct from objects with the same names outside that function-definition or within other invocations of that function-definition."

```
670     CASE "NEIN", "nein"
680           ANTWORT = 2
690     CASE ELSE
700           ANTWORT = 3
710     END SELECT
720 LOOP WHILE ANTWORT = 3
730 END FUNCTION
```

Im vorstehenden Programm wird die Variable A$ implizit als einfache alphanumerische Variable dimensioniert und steht in keinerlei Beziehung zu einer eventuell gleichlautenden Variablen im Hauptprogramm.
Der Aufruf einer externen Funktion geschieht auf dieselbe Weise wie der eines DEF-Blocks.[221]

333.2 Programmsegmentierungsanweisungen

Aufbau und Wirkungsweise der Programmsegmentierungsanweisungen sind in Bd. 2, Abs. 333.2 ausführlich beschrieben; durch ANSI BASIC 82 werden keine Normänderungen vorgesehen.[222]

334 Graphikanweisungen

Graphikanweisungen dienen zur Erzeugung von Bildern und Zeichnungen auf peripheren graphischen Geräten.[223]
Unter den graphischen Geräten erlangen – neben den wohl am häufigsten benutzten Plottern – Matrixdrucker und graphische Terminals in zunehmendem Masse Bedeutung. Diese Geräte erlauben teilweise auch die Erstellung mehrfarbiger Bilder.[224]
Bei allen Geräten wird die Bild- oder Zeichenfläche in Zeilen und Spalten aufgeteilt, wobei jeder Quadrant normalerweise zur Ausgabe eines Zeichens dient. Bezüglich der einzelnen Geräte bestehen jedoch wesentliche Unterschiede.
Beispielsweise unterteilt sich der Bildschirm des Comodore 2001 in 24

221 Vgl. Bd. 1, Abs. 325.12, 325.312 und 325.322.
222 Vgl. ANSI BASIC 82, p. 82–83.
223 ANSI BASIC 79, p. 105: "Graphical facilities are provided for producing two-dimensional graphs and drawings on graphic display devices such as a pen and ink plotter, a vector cathode ray tube, or a raster cathode ray tube." – ANSI BASIC 82, p. 156.
224 Hans Kürner. Peripheriegeräte. In: Karl Steinbach und Wolfgang Weber. Taschenbuch der Informatik 2, Struktur und Programmierung von EDV-Systemen. Berlin 1974, S. 465–479.

Zeilen mit jeweils 40 Spalten, also 960 Quadranten, wobei jeder Quadrant aus 8 × 8 Bildpunkten besteht. Je nach Art des darzustellenden Bildes werden aus den 64 Punkten eines Quadranten die notwendigen Bildpunkte eingeblendet. Zur Bilderstellung stehen insgesamt über 60000 Einzelpunkte zur Verfügung.[225]

Bei entsprechender Programmierung kann jeder Bildschirmpunkt einzeln angesteuert werden und damit zur Erzeugung eines Bildes verwendet werden. Die maximale Auflösung richtet sich nach der Feinheit der Rasterung des graphischen Terminals.

Auch mechanische Zeichengeräte – Plotter und Matrixdrucker – weisen eine spezifische Rasterstruktur auf, die jeweils durch den minimalen Abstand zwischen zwei Bildpunkten bestimmt wird.

Die Verwendung von Graphikanweisungen ist nur im Rahmen besonders leistungsfähiger BASIC-Programmiersysteme möglich. Ansatzpunkte zu ihrer Standardisierung wurden bereits 1974 von Arthur Luehrmann entwickelt[226] und in der Folge weitgehend in die Normvorschläge von ANSI 'ASIC übernommen.[227]

ie Statemententwicklung von ANSI BASIC 79 zu ANSI BASIC 82 lässt ·kennen, dass sich bereits ein Nukleus von Statementformen gebildet hat. ıer in seiner Wirkungsweise unverändert geblieben ist und teilweise nur geringfügige Kennwortänderungen erfahren hat. Um dem Leser einen Überblick zur Normentwicklung der Graphikanweisungen zu verschaffen, sind die beiden Normvorschläge ANSI BASIC 79 und ANSI BASIC 82 in den folgenden Abschnitten gegenübergestellt.

Die Graphikanweisungen lassen sich generell in vier Gruppen unterteilen; vgl. Abbildung 12.

⟨Graphikanweisung⟩ : = ⟨Koordinatenanweisung⟩ | ⟨Bildpunktanweisung⟩ | ⟨Zeichenanweisung⟩ | ⟨Bildunterprogrammanweisung⟩

318 206 319 332 340 354

225 Commodore. PET 2001. Angebotsprospekt. Neu-Isenburg 1979. – Das Farb-Datensichtgerät IBM 3279, Modell B03, verfügt über 32 Zeilen zu 80 Spalten mit einem 9 × 12-Punktraster bei 7 Farben, IBM Form GT R-2697-2 [Stuttgart 1981].

226 Arthur Luehrmann. An Elaboration of Some Thoughts on a Graphical Syntax in BASIC. Hanover 1974.

227 ANSI BASIC 79, p. 105–118. – ANSI BASIC 82, p. 156–173.

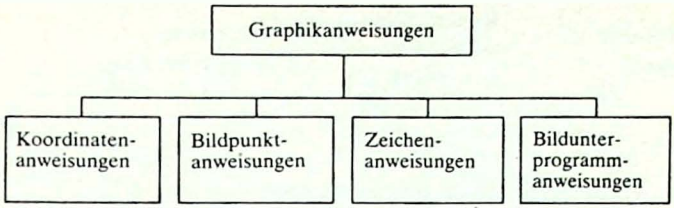


Abbildung 12.

Über die Graphikanweisungen lässt sich das zu verwendende Koordinatensystem definieren; ausserdem können Art und Zeichenzustand von Bildpunkten festgelegt oder abgefragt werden; spezielle Zeichenanweisungen beziehen sich auf die Graphikherstellung im engeren Sinne, insbesondere auf das Zeichnen von Punkten und Linien sowie die Eingabe und Abfrage von aktuellen Bildpunktpositionen über das graphische Terminal; weiterhin können Graphikanweisungen zu Bildunterprogramm zusammengefasst werden.
Die zur Graphikbearbeitung erforderlichen BASIC-Anweisungen bewirken im allgemeinen kombinierte Operationen (compound graphical operations)[228], was im allgemeinen eine relativ komplexe Grundstruktur erfordert.
Die folgenden Ausführungen beziehen sich weitgehend auf die Normvorschläge von ANSI BASIC.[229]

334.1 Koordinatenanweisungen

Die Koordinatenanweisungen dienen zur Definition und Abfrage von Bild-, Zeichenflächen- und Problemkoordinaten.[230] Dementsprechend können auch verschiedene Statementtypen unterschieden werden, die sich zu Gruppen von Definitions- und Abfrageanweisungen zusammenfassen lassen, vgl. Abbildung 13.

228 Vgl. hierzu auch Luehrmann, An Elaboration of Some Thoughts on a Graphical Syntax in BASIC, p. 4–5: "A *compound* graphical operation is any time-ordered set of elementary graphical operations. ... An *elementary* graphical operation is one which establishes the value of a *single element* of the device state vector, without affecting the remaining elements."
229 ANSI BASIC 79, p. 105–118. – ANSI BASIC 82, p. 156–173.
230 ANSI BASIC 79, p. 105: "Coordinate systems." – ANSI BASIC 82, p. 156.

⟨Koordinaten-anweisung⟩ 319 : = ⟨Koordinaten-definitions-anweisung⟩ 320 | ⟨Koordinaten-abfrageanweisung⟩ 326 318

334.11 Koordinatendefinitionsanweisungen

Die Koordinatendefinitionsanweisungen beziehen sich zunächst auf die Definition der physikalischen Bildfläche, die auf dem zu verwendenden graphischen Ausgabegerät zur Verfügung stehen soll.[231] Bei einem Datensichtgerät ist dies die Bildschirmgrösse und bei einem Plotter die maximal bearbeitbare Papiergrösse.
Weiterhin wird über die Koordinatendefinitionsanweisungen die Zeichenfläche festgelegt. Diese ist ein Quadrant innerhalb der definierten physikalischen Bildfläche. Innerhalb der Zeichenfläche wird die graphische Ausgabe erzeugt; die Zeichenfläche kann demnach maximal die physikalische Bildfläche umfassen.[232]
Um für die Zeichenfolge ein problemorientiertes Bezugssystem zu erhalten, muss ein System von Problemkoordinaten angegeben werden;[233] hierzu dienen spezielle Problemkoordinatenanweisungen.
Für die Koordinatendefinitionsanweisungen existieren drei verschiedene Statements:

⟨Koordinaten-definitionsanweisung⟩ 320 : = ⟨SET BOUNDS-Statement⟩ 321 | ⟨SET VIEWORT-Statement⟩ 324 | ⟨SET WINDOW-Statement⟩ 325 319

231 ANSI BASIC 79, p. 105: "The largest rectangular area in which graphic output or input can occur on a graphic device is called the screen. Positions on this screen are described by a system of screen coordinates." – ANSI BASIC 82, p. 156.

232 ANSI BASIC 79, p. 105: "Graphic output may be located anywhere on the screen and need not occupy the entire screen. A rectangular 'viewport' on the screen, in which all output is to occur, may be specified by a [SET VIEWPORT-Statement]." – ANSI BASIC 82, p. 156.

233 ANSI BASIC 79, p. 105: "The coordinates used to produce graphic output inside the viewport on the screen may be chosen to suit the application." – ANSI BASIC 82, p. 156: "The range of this system of 'problem coordinates' is established by a set-statement."

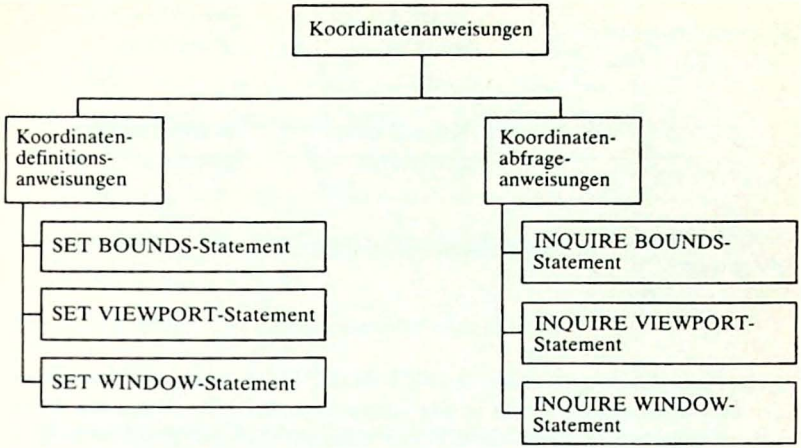


Abbildung 13

334.111 SET BOUNDS-Statement
Mit Hilfe des SET BOUNDS-Statements werden die physikalischen Bildflächenkoordinaten festgelegt. Es weist folgende Struktur auf:[234]

⟨SET BOUNDS-Statement⟩ : =
321 320

SET BOUNDS ⟨Koordinatengrenzliste⟩ 322

Die Koordintatengrenzliste besteht immer aus vier – durch Kommata voneinander getrennten – Koordinatengrenzwerten, wobei jeder dieser Grenzwerte über einen arithmetischen Ausdruck spezifiziert werden kann.

234 ANSI BASIC 79, p. 105.

⟨Koordinatengrenzliste⟩ : = ⟨Koordinaten-grenzwert⟩ , ⟨Koordinaten-grenzwert⟩ ,
322 321 | 324 | 325 323 323

⟨Koordinaten-grenzwert⟩ , ⟨Koordinaten-grenzwert⟩ ,
323 323

⟨Koordinaten-grenzwert⟩ : = ⟨Arithmetischer Ausdruck⟩
323 322 49

In der Koordinatengrenzliste des SET BOUNDS-Statements bezeichnen die Koordinatengrenzwerte in der angegebenen Reihenfolge die linke, rechte, untere und obere Grenze der genutzten physikalischen Bildfläche. Soll die maximal zulässige physikalische Bildfläche benutzt werden, so kann das SET BOUNDS-Statement entfallen.[235]
Bei den in der Koordinatengrenzliste angeführten Werten handelt es sich nach ANSI BASIC nicht um physikalische Grössen, sondern um relative Werte, die als Bildeinheiten (BE, screen units) bezeichnet werden. Implizit soll die kürzere Seite der physikalischen Bildfläche in hundert Einheiten eingeteilt sein; die längere Seite ist entsprechend grösser.[236]
Die Normänderung nach ANSI BASIC 82 sieht vor, dass anstelle von Koordinatengrenzwerten die relative Höhe und Breite im SET BOUNDS-Statement angegeben werden.[237] Die längere Seite der beiden Werte soll eine Bildschirmeinheit lang sein.[238]

235 ANSI BASIC 79, p. 105: "[The system of screen coordinates] is established initially by default and may be changed by a statement SET BOUNDS LO, RO, BO, TO where LO, RO, BO, and TO are numeric-expressions assigning coordinates to the left, right, bottom, and top edges of the screen."
236 ANSI BASIC 79, p. 106: "The physical location of the output on the screen is determined by a mapping between problem coordinates and screen coordinates specific to the display device. At the start of th program execution the screen coordinate system shall be set by default so that its origin (0,0) is at the lower left corner of the physical screen, the shorter of the two sides of the screen is 100 screen units long, and the other side is proportionately longer."
237 ANSI BASIC 82, p. 157: "Execution of a set-statement with the keyword BOUNDS shall establish the relative height and width of the assumed screen."
238 ANSI BASIC 82, p. 157: "The height and width must both be greater than zero and less than or equal to one. At least one of the height and width must equal one."

⟨SET BOUNDS-Statement⟩ : =
321 320

SET BOUNDS ⟨Relative Höhe⟩ , ⟨Relative Breite⟩

⟨Relative Höhe⟩ : = ⟨Arithmetischer Ausdruck⟩
49

⟨Relative Breite⟩ : = ⟨Arithmetischer Ausdruck⟩
49

Beide Werte von relativer Höhe und Breite müssen kleiner/gleich 1 und grösser als Null sein. Einer der beiden Werte muss 1 sein.[239]

334.112 *SET VIEWPORT*-Statement
Die zu benutzende Zeichenfläche wird über das

⟨SET VIEWPORT-Statement⟩ : =
324 320

SET VIEWPORT ⟨Koordinatengrenzliste⟩ 322

festgelegt.
Die Angaben in der Koordinatengrenzliste erfolgen in "screen units" (BE). Die Reihenfolge der Grenzwertangabe bezeichnet nacheinander die linke, rechte, untere und dann obere Ecke der Zeichenfläche.[240]
Die einzelnen Koordinatengrenzwerte des SET VIEWPORT-Statements dürfen nicht die im SET BOUNDS-Statement implizit oder explizit festgelegten Grenzwerte überschreiten.

239 ANSI BASIC 82, p. 157: "The longer of the height and width of the screen is 1 screen unit long, and the other dimension is proportionally shorter."
240 ANSI BASIC 79, p. 106: "The VIEWPORT boundaries shall specify the screen coordinates of the left, right, bottom, and top edges of the viewport." – ANSI BASIC 82, p. 157.

334.113 SET WINDOW-Statement

Durch das SET WINDOW-Statement werden die Grenzen der Problemkoordinaten bestimmt.

⟨SET WINDOW-Statement⟩ : =
325 320

SET WINDOW ⟨Koordinatengrenzliste⟩ 322

Die Angaben in der Koordinatengrenzliste basieren auf einer frei wählbaren Mass-Dimension (m, kg, etc.) in Problemeinheiten (PE).
Ein korrespondierendes Paar von Zeichenflächen- und Problemkoordinatenstatements (SET VIEWPORT- und SET WINDOW-Statements) bildet die Grundlage zur Berechnung des Transformationsmassstabes; er tellt eine lineare Beziehung zwischen Bildschirm- und Problemeinheiten ıer.[241] Physikalisch stimmen die Grenzen der Problemkoordinaten mit Jenen der durch das SET VIEWPORT festgelegten Zeichenfläche überein.
Als Beispiel soll auf einem Bildschirm mit maximaler Höhe von 30 cm und maximaler Breite von 20 cm eine Zeichenfläche ausgenutzt werden, die einen umlaufenden Rand von 1,5 cm hat. In diese Zeichenfläche (27 cm x 17 cm) soll eine Graphik übertragen werden, die den Stromverbrauch eines Einfamilienhauses während 30 Tagen angibt. Kann der Stromverbrauch maximal 100 Kilowatt betragen, so sind Bildschirmfläche, Zeichenfläche und Problembezugssystem mit den entsprechenden Koordinatenanweisungen nach ANSI BASIC 79 wie folgt zu definieren:

```
100 SET BOUNDS 0,100, 0,30/20*100
110 SET VIEWPORT 15,85, 15,30/20*100-15
120 SET WINDOW 0,30, 0,100
```

Nach ANSI BASIC 82 müsste angegeben werden:

```
100 SET BOUNDS 1, 20/30
110 SET VIEWPORT 0.05, 17/30, 0.05, 0.9
120 SET WINDOW 0, 30, 0, 100
```

241 ANSI BASIC 79, p. 106: "The mapping between problem coordinates and screen coordinates is established by two set-statements: those with the keywords WINDOW and VIEWPORT. This mapping, known as the windowing transformation, shall be determined by the requirement that it be linear and that the WINDOW boundaries (specified in problem coordinates) map onto the VIEWPORT boundaries (specified in screen coordinates)." – ANSI BASIC 82, p. 157.

Bei Koordinatendefinitionsanweisungen treten Fehler auf, wenn sich logisch widersprechende Koordinatengrenzen angegeben werden[242] oder wenn sich für eine Koordinatenrichtung eine Weite von Null ergibt.[243] Nach ANSI BASIC 82 sollte die Ausführungsfolge der Koordinatendefinitionsanweisungen stets in der Reihenfolge SET BOUNDS-Statement vor SET VIEWPORT und vor SET WINDOW-Statement festgelegt werden. Ohne logisch vorausgehende Koordinatendefinitionsanweisung darf keine graphische Anweisung angegeben werden.[244]
Generell gelten für einen Bildpunkt mit den Koordinaten (X, Y) die folgenden Beziehungen:

$$ZX = \frac{ZR-ZL}{PR-PL} \quad (PX-PL)$$

$$ZY = \frac{ZO-ZU}{PO-PU} \quad (PY-PU)$$

$$BX = BL + ZL + ZX$$
$$BY = BU + ZU + ZY$$

mit

BL = Linke Bildflächengrenze in BE (Bildeinheiten/screen units)
BR = Rechte Bildflächengrenze in BE
BU = Untere Bildflächengrenze in BE
BO = Obere Bildflächengrenze in BE
BX = X-Koordinate eines Bildpunktes in BE
BY = Y-Koordinate eines Bildpunktes in BE
ZL = Linke Zeichenflächengrenze in BE
ZR = Rechte Zeichenflächengrenze in BE

242 ANSI BASIC 79, p. 107: "The orientation of the boundaries specified in a viewport-statement differs from that of the screen; e. g., the right coordinate of the viewport is less than the left coordinate, while the coordinate of the right edge of the screen is greater than the coordinate of the left edge."
243 ANSI BASIC 79, p. 107: "The boundaries in a set-statement specify a rectangle of zero width or hight." – ANSI BASIC 82, p. 158.
244 ANSI BASIC 82, p. 157: "A set-statement with the keyword BOUNDS may be executed at most once during the execution of a program and must not be executed following execution of a set-statement with the keywords VIEWPORT or WINDOW nor following execution of any statement which produces graphical output."

ZU = Untere Zeichenflächengrenze in BE
ZO = Obere Zeichenflächengrenze in BE
ZX = X-Koordinate eines Bildpunktes in bezug auf die Zeichenfläche mit dem Ursprung (ZL, ZU) in BE
ZY = Y-Koordinate eines Bildpunktes in bezug auf den Zeichenflächenursprung (ZL, ZU) in BE
PL = Linke Grenze des Problemkoordinatensystems in PE (Problemeinheiten/problem units)
PR = Rechte Grenze des Problemkoordinatensystems in PE
PU = Untere Grenze des Problemkoordinatensystems in PE
PO = Obere Grenze des Problemkoordinatensystems in PE
PX = X-Koordinate eines Bildpunktes in PE
PY = Y-Koordinate eines Bildpunktes in PE

Vgl. hierzu auch Abbildung 14

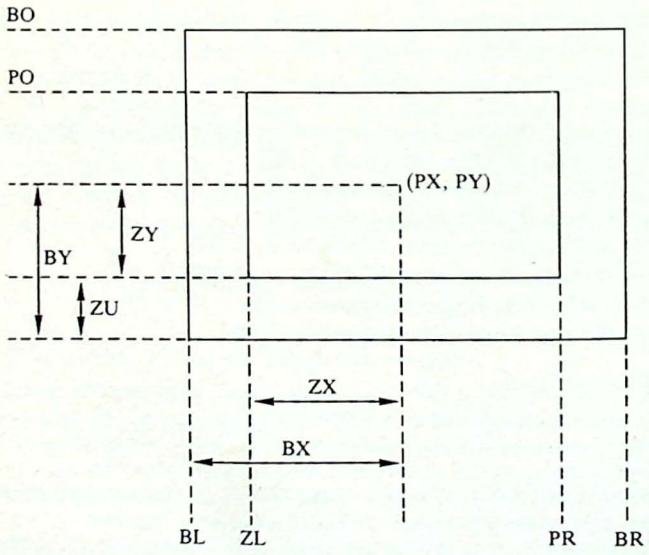


Abbildung 14. Beziehung der Bildflächen-, Zeichenflächen- und Problemkoordinatensysteme

334.12 Koordinatenabfrageanweisungen

Den Koordinatendefinitionsanweisungen stehen entsprechende Koordinatenabfrageanweisungen gegenüber.[245] Sie dienen generell zur Ermittlung der via SET BOUNDS-, SET VIEWPORT- und SET WINDOW-Statements festgelegten Koordinatengrenzwerte.[246]
Für die Koordinatenabfrageanweisungen gilt gemäss Abbildung 13:

⟨Koordinaten-abfrageanweisung⟩ (326) : = (319) ⟨INQUIRE BOUNDS-Statement⟩ (327) |

⟨INQUIRE VIEWPORT-Statement⟩ (328) |

⟨INQUIRE WINDOW-Statement⟩ (329)

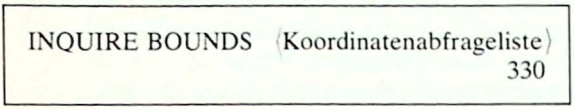
Nach ANSI BASIC 79 stimmen die drei Statements in ihrer Grundform überein; sie enthalten eine Koordinatenabfrageliste, die aus vier (durch Kommata voneinander getrennten) numerischen Variablen besteht. Diese Variablen erhalten bei einer Abfrage – in Analogie zur Koordinatengrenzliste – den fixierten linken, rechten, unteren und oberen Koordinatengrenzwert.
Nach ANSI BASIC 82 weicht nur die Form des INQUIRE BOUNDS-Statement davon ab. Ausserdem wird an die Stelle des Kennworts INQUIRE nunmehr bei allen Koordinatenabfrageanweisung das Kennwort ASK gesetzt.[247]

334.121 INQUIRE BOUNDS-Statement

Durch das INQUIRE BOUNDS-Statement werden die bestehenden Bildflächengrenzwerte abgefragt.

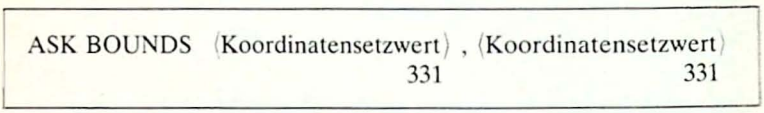
245 ANSI BASIC 79, p. 105: "An inquire-statement is provided to determine the current values of the parameters established by execution of one of these set-statements." – ANSI BASIC 82, p. 156.
246 ANSI BASIC 79, p. 107: "Execution of an inquire-statement with one of the keywords BOUNDS, VIEWPORT, or WINDOW shall provide information about the current values of the boundaries for the specified rectange." – ANSI BASIC 82, p. 158.
247 ANSI BASIC 82, p. 156–157.

⟨INQUIRE BOUNDS-Statement⟩ : =
327 326



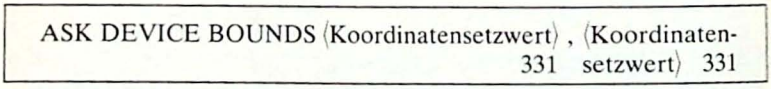
Nach ANSI BASIC 82 können als Koordinatengrenzwerte im INQUIRE BOUNDS-Statement nur die relative Höhe und Breite abgefragt werden.[248]

⟨INQUIRE BOUNDS-Statement⟩ : =
327 326



Hinzugekommen ist eine Erweiterung, mit der die Maximalwerte für relative Höhe und Breite des benutzten graphischen Ausgabegerätes erfragt werden können.[249] ANSI BASIC 82 schreibt hierfür die nachstehende Form vor:

⟨INQUIRE DEVICE BOUNDS-Statement⟩ : =



334.122 INQUIRE VIEWPORT-Statement
Das INQUIRE VIEWPORT-Statement dient zur Abfrage der Zeichenflächengrenzwerte.

248 ANSI BASIC 82, p. 156.
249 ANSI BASIC 82, p. 158: "Execution of an ask-statement with the keywords DEVICE BOUNDS shall assign to the numeric-variables the relative height and width of the physical screen. That is, the longer of the height and width shall have a value of 1 and the shorter a proportionately smaller value."

⟨INQUIRE VIEWPORT-Statement⟩ : =
328 326

INQUIRE VIEWPORT ⟨Koordinatenabfrageliste⟩ |
330

ASK VIEWPORT ⟨Koordinatenabfrageliste⟩
330

334.123 INQUIRE WINDOW-Statement
Mit Hilfe des INQUIRE WINDOW-Statements können die aktuellen Problemkoordinatengrenzwerte ermittelt werden.

⟨INQUIRE WINDOW-Statement⟩ : =
329 326

INQUIRE WINDOW ⟨Koordinatenabfrageliste⟩ |
330

ASK WINDOW ⟨Koordinatenabfrageliste⟩
330

Den in der Koordinatenabfrageliste erscheinenden numerischen Variablen werden bei Ausführung der genannten Statements die gesetzten Koordinatengrenzwerte zugewiesen; sie werden in Bild- resp. Problemeinheiten ausgedrückt.

⟨Koordinatenabfrageliste⟩ : = ⟨Koordinatensetzwert⟩ , ⟨Koordinatensetzwert⟩ ,
330 327 | 328 | 329 331 331

⟨Koordinatensetzwert⟩ , ⟨Koordinatensetzwert⟩
331 331

⟨Koordinatensetzwert⟩ : = ⟨Numerische Variable⟩
331 330 36

334.2 Bildpunktanweisungen

Das auf einem graphischen Gerät[250] erzeugte Bild setzt sich aus einer Menge von Punkten oder Punktverbindungen in Form von Linien zusammen.
Das Medium zum Zeichnen eines Punktes oder einer Linie wird im folgenden Bildpunkt (beam) genannt. Beim graphischen Terminal handelt es sich um einen Elektronenstrahl, beim Plotter um einen elektromechanischen Stift.[251]
Bildpunktanweisungen dienen dazu, Art und Zustand eines Zeichenmediums festzulegen oder abzufragen. Dementsprechend kann zwischen Bildpunktangabe- und Bildpunktabfrageanweisungen unterschieden werden; vgl. Abbildung 15.

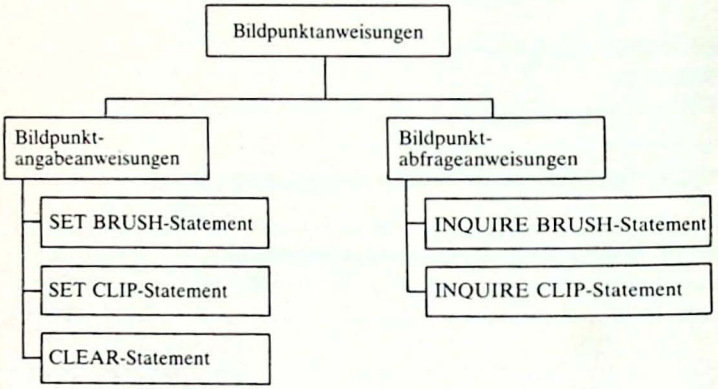


Abbildung 15.

⟨Bildpunkt-anweisung⟩ : = ⟨Bildpunkt-angabeanweisung⟩ | ⟨Bildpunkt-abfrageanweisung⟩
332 318 333 337

250 Vgl. Abs. 333.2, Einleitung.
251 ANSI BASIC 79, p. 108: "A mechanism for drawing lines on the screen of a graphic display device is called a beam. In reality, a beam may be an ink pen on a plotter, an electron beam of a vector cathode ray tube, or a pointer to the memory map of a raster cathode ray tube." – ANSI BASIC 82, p. 160.

334.21. Bildpunktangabeanweisungen

Die Bildpunktangabeanweisungen lassen sich in drei Statements einteilen.

⟨Bildpunkt-angabeanweisung⟩ 333 : = ⟨SET BRUSH-Statement⟩ 334 | ⟨SET CLIP-Statement⟩ 335 | ⟨CLEAR-Statement⟩ 336

334.211 SET BRUSH-Statement

Mit dem SET BRUSH-Statement wird die Einstellung eines Bildpunktes (beam style) durch Angabe entsprechender Attribute spezifiziert. Insbesondere können Vorgaben bezüglich Farbe, Helligkeit, Breite oder Art der Linienführung gemacht werden.[252]

Die Spezifikation der Bildpunkteinstellung erfolgt im SET BRUSH-Statement durch Angabe einer systemspezifisch zulässigen Ganzzahl.[253]

Die Festlegung der zur Bildpunkteinstellung zu verwendenden Nummern ist nach ANSI BASIC implementationsabhängig;[254] Einstellungsart 1 soll den Normalzustand eines Bildpunktes initiieren.[255]

⟨SET BRUSH-Statement⟩ 334 : = 333

SET BRUSH	|	
SET BRUSH	⟨Arithmetischer Ausdruck⟩	49

252 ANSI BASIC 79, p. 108: "A graphic display device may possess several styles of beams, each with a particular color, intensity, width, or line texture." – ANSI BASIC 82, p. 160.

253 ANSI BASIC 79, p. 109: "Execution of a set-statement with the keyword BRUSH shall cause the index to be evaluated to obtain an integer N and shall establish the style of the beam for subsequent line drawing to be the Nth one of the set of available beam styles."

254 ANSI BASIC 79, p. 109: "The number of beam styles available is implementation-defined, as are the attributes (e.g., color, intensity, width, and line texture) of each style."

255 ANSI BASIC 79, p. 109: "At the initiation of program execution, the beam style shall be one. A beam style of zero, if available, shall correspond to a beam whose color and brightness are the same as the background of the display screen."

Der im vorerwähnten Statement erscheinende arithmetische Ausdruck wird durch Runden ganzzahlig gemacht.[256] Fehlt eine explizite Einstellungsangabe, so wird Einstellungsart 1 realisiert.[257]
ANSI BASIC 82 löst das SET BRUSH-Statement in drei einzelne Statementformen auf, mit denen die Art des Linienzugangs, die Vordergrund- und die Hintergrundfarbe festgelegt werden.[258]

⟨SET BRUSH-Statement⟩ : =
334 333

SET LINE STYLE ⟨Arithmetischer Ausdruck⟩ |
49

SET COLOR ⟨Arithmetischer Ausdruck⟩ |
49

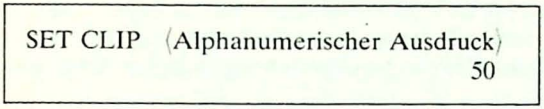
SET BACKGROUND COLOR ⟨Arithmetischer Ausdruck⟩
49

Die Vordergrundfarbe unterscheidet sich von der Hintergrundfarbe durch einen höheren Helligkeitsgrad.

334.212 SET CLIP-Statement
Mit dem SET CLIP-Statement wird der sichtbare Zeichenbereich eines Bildpunktes über die Schalterstellungen "ON" oder "OFF" festgelegt. Falls der Schalter "ON" ist, erfolgt die Graphikausgabe in der gewählten Bildpunkteinstellung nur dann, wenn der Bildpunkt durch eine Zeichenanweisung[259] innerhalb der via SET VIEWPORT-Statement festgelegten Zeichenfläche bewegt wird. Ist der Schalter "OFF", so entfällt diese Begrenzung und es wird die gesamte physikalisch verfügbare Bildfläche für die Graphikausgabe freigegeben.[260]

256 ANSI BASIC 79, p. 108.
257 ANSI BASIC 79, p. 108: "The default beam style is brush one."
258 ANSI BASIC 82, p. 161.
259 Vgl. Abs. 334.3.
260 ANSI BASIC 79, p. 108: "The beam has two states. When it is on, lines are drawn as it changes position. ... When it is off, no lines are drawn as it changes position. Ordinarily no graphic output is displayed outside the viewport on the screen." – Vgl. auch die noch exakteren Ausführungen in ANSI BASIC 79.

⟨SET CLIP-Statement⟩ : =
335 333

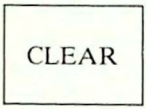


Der Inhalt des alphanumerischen Ausdrucks darf nur "ON" oder "OFF" sein. Bei Programmbeginn wird der Zeichenzustand eines Bildpunktes implizit auf "ON" gesetzt.[261] Mithin ist das SET CLIP-Statement nur dann zu verwenden, wenn die via SET VIEWPORT-Statement festgelegte Zeichenfläche während des Programmablaufs temporär überschritten werden soll.[262]

334.213 CLEAR-Statement

Durch das CLEAR-Statement wird der gesamte Bildschirm eines graphischen Terminals gelöscht, der Bildpunkt in seinen Ausgangszustand zurückversetzt und ausgeschaltet.[263] Bei einem Plotter wird die Papierzeichenfläche freigegeben, so dass der Benutzer eventuell neues Papier einlegen kann.

⟨CLEAR-Statement⟩ : =
336 330



p. 109: "Execution of the set-statement with the keyword CLIP shall enable clipping or disable it depending on whether the value of the string-expression is 'ON' or 'OFF'. When clipping is enabled, no graphic output shall occur outside the viewport. ...; when it is disabled, all graphic output that falls within the boundaries of the screen shall be displayed, whether or not it falls within the viewport." – ANSI BASIC 82, p. 162.

261 ANSI BASIC 79, p. 109: "At the initiation of program execution clipping shall be enabled." – ANSI BASIC 82, p. 162.

262 Vgl. auch ANSI BASIC 79, p. 108: "The statements SET CLIP 'OFF' and SET CLIP 'ON' may be used to disable [the clipping] feature and to enable it again."

263 ANSI BASIC 79, p. 108: "The clear-statement ... clears the entire screen, returning it to its background color, and also turns the beam off." – ANSI BASIC 82, p. 160.

Bei Bildpunktangabeanweisungen bestehen folgende Fehlermöglichkeiten:[264]

- Der ganzzahlige Wert im SET BRUSH-Statement ist negativ oder es existiert keine dazu passende Punkt- oder Farbart.
- Im SET CLIP-Statement wird die Schalterstellung nicht mit "ON" oder "OFF" angegeben.

334.22 Bildpunktabfrageanweisungen

Mit den Bildpunktabfrageanweisungen können Art und Zustand eines Bildpunktes erfragt und die entsprechenden Werte in Variable übertragen werden.[265]

⟨Bildpunkt-abfrageanweisung⟩ (337) : = ⟨INQUIRE BRUSH-Statement⟩ (338) | ⟨INQUIRE CLIP-Statement⟩ (339)

Mit ANSI BASIC 82 treten im INQUIRE BRUSH-Statement die zum SET BRUSH-Statement[266] analogen Verfeinerungen ein. Ebenso ist das Kennwort INQUIRE durch ASK substituiert.

334.221 INQUIRE BRUSH-Statement

Das INQUIRE BRUSH-Statement dient zur Abfrage der gewählten Punktart. Es weist nach ANSI BASIC 79 die Form:

⟨INQUIRE BRUSH-Statement⟩ (338) : = (337)

INQUIRE BRUSH ⟨Numerische Variable⟩ (36)

auf.[267]

264 ANSI BASIC 79, p. 109: "The beam style selected by a set-statement with the keyword BRUSH is not available. ... The value of the string-expression in a set-statement with the keyword CLIP is neither 'ON' nor 'OFF'."

265 ANSI BASIC 79, p. 108: "The current style of the beam and current state of clipping may be determined by the inquire-statements."

266 Vgl. Abs. 334.211.

267 ANSI BASIC 79, p. 109: "Execution of an inquire-statement with the keyword

Für das INQUIRE BRUSH-Statement können nach ANSI BASIC 82 folgende Formen angegeben werden:[268]

⟨INQUIRE BRUSH-Statement⟩ : =
338 337

```
ASK LINE STYLE ⟨Numerische Variable⟩ |
                                    36

ASK COLOR ⟨Numerische Variable⟩ |
                                 36

ASK BACKGROUND COLOR ⟨Numerische Variable⟩ |
                                              36

ASK COLOR COUNT ⟨Numerische Variable⟩ |
                                       36

ASK STYLE COUNT ⟨Numerische Variable⟩
                                       36
```

Durch die Abfrage mit dem ASK COLOR COUNT-Statement wird die grösstmögliche Farbzahl für ein SET COLOR-Statement angegeben.[269] Analog kann mit dem ASK STYLE COUNT-Statement die mögliche Anzahl an Linienarten des vorhandenen Graphiksystems ermittelt werden.[270]

BRUSH shall assign the number of the current beam style to the numeric-variable."

268 ANSI BASIC 82, p. 160–161.

269 ANSI BASIC 82, p. 162: "Execution of an ask-statement with the keyword COLOR COUNT shall assign to the numeric-variable the largest distinct value of COLOR available."

270 ANSI BASIC 82, p. 161: "Execution of an ask-statement with the keywords STYLE COUNT shall assign to the numeric-variable the largest value of LINE STYLE available."

334.222 INQUIRE CLIP-Statement

Das INQUIRE CLIP-Statement ermittelt die Art der aktuellen Zeichenflächenbegrenzung eines Bildpunktes.[271]

⟨INQUIRE CLIP-Statement⟩ : =
339 338

INQUIRE CLIP ⟨Alphanumerische Variable⟩ |
44

ASK CLIP ⟨Alphanumerische Variable⟩
44

Nach ANSI BASIC korrekte Bildpunktanweisungen stellen die nachfolgenden Statements dar.

```
220 SET BRUSH 4
330 SET BRUSH X(N)

150 SET CLIP "OFF"
190 SET CLIP A$

300 INQUIRE BRUSH N      ! ASK BRUSH N

950 INQUIRE CLIP F$      ! ASK CLIP F$
```

Mit dem ersten Statement wird ein Bildpunkt der Art 4 (eventuell eine bestimmte Farbe) initialisiert; mit dem zweiten Statement wird der ganze physikalische Zeichenbereich zur Graphikausgabe freigegeben. Die beiden letzten Statements fragen die aktuelle Bildpunktart und die Art der Zeichenflächenbegrenzung ab.

334.3 Zeichenanweisungen

Zeichenanweisungen dienen zur Bewegung des Bildpunktes mit oder ohne graphischer Ausgabe auf der Bildfläche sowie zur Abfrage der aktuellen Bildpunktposition.
Bezüglich der Zeichenanweisungen kann generell zwischen Ausgabe-,

271 ANSI BASIC 82, p. 162: "Execution of an ask-statement with the keyword CLIP shall assign the value 'ON' to the string-variable if clipping is enabled and the value 'OFF' if it is disabled. The values returned shall be all upper-case letters." – ANSI BASIC, p. 109.

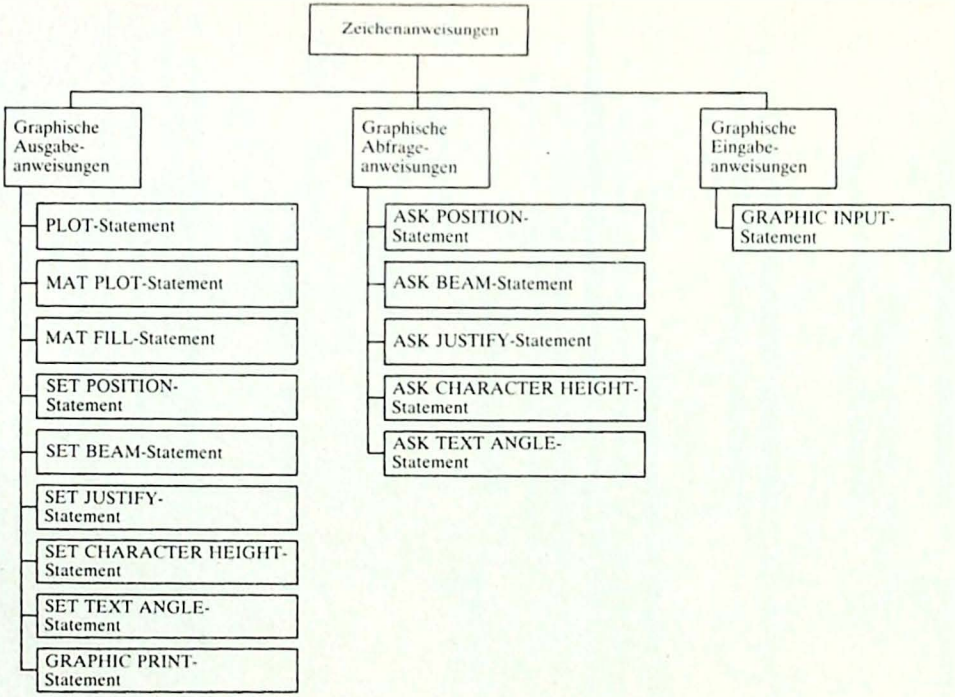


Abbildung 16.

Abfrage- und Eingabeanweisungen differenziert werden; vgl. Abbildung 16, aus der auch die nach ANSI BASIC vorgesehenen – dort aber noch nicht im vorerwähnten Sinne strukturierten – Zeichenanweisungen ersichtlich sind.

⟨Zeichenanweisung⟩ : = ⟨Graphische Ausgabeanweisung⟩ |
340 318 341

⟨Graphische Abfrageanweisung⟩ |
357

⟨Graphische Eingabeanweisung⟩
363

334.31 Graphische Ausgabeanweisungen

Für die nach ANSI BASIC genormten graphischen Ausgabeanweisungen gilt:[272]

⟨Graphische Ausgabeanweisung⟩ : = ⟨PLOT-Statement⟩ |
341 340 342

⟨MAT PLOT-Statement⟩ |
346

⟨MAT FILL-Statement ⟩ |
350

⟨SET POSITION-Statement⟩ |
351

⟨SET BEAM-Statement⟩ |
352

⟨SET JUSTIFY-Statement⟩ |
353

272 ANSI BASIC 82, p. 163-169.

⟨SET CHARACTER HEIGHT-Statement⟩ |
354

⟨SET TEXT ANGLE-Statement⟩ |
355

⟨GRAPHIC PRINT-Statement⟩
356

Davon sind von ANSI BASIC 79 nur das PLOT-, SET POSITION- und das GRAPHIC PRINT-Statement genormt.[273]

334.311 PLOT-Statement

Das PLOT-Statement dient zur Generation graphischer Ausgaben auf entsprechenden peripheren Geräten.[274]

Syntax und Wirkungsweise des PLOT-Statements entsprechen nach ANSI BASIC den Regeln des PRINT-Statements.[275] Für die Syntax gilt daher:

⟨PLOT-Statement⟩ : =
342 341

PLOT |
PLOT ⟨Plotliste⟩
343

Die Plotliste enthält eine Menge von Koordinatenpaaren, die durch Komma oder Semikolon voneinander getrennt sind. Die zur Spezifikation eines Koordinatenpaares erforderlichen arithmetischen Ausdrücke sind ihrerseits durch ein Komma separiert und durch Klammern zusam-

273 ANSI BASIC 79, p. 111–114.

274 ANSI BASIC 79, p. 111: "Graphic output is generated by the plot-statement." – ANSI BASIC 82, p. 163: "Line segments and points are drawn by the plot-statement and the array-plot-statement."

275 ANSI BASIC 79, p. 111: "[The plot-statement] is analogous to the print-statement and... can be used to draw points and line segments." – ANSI BASIC 82, p. 163.

mengefasst. Ausserdem kann die Plotliste mit einem Semikolon bzw. Komma abgeschlossen werden.[276]

⟨Plotliste⟩ 343 342 : = ⟨Koordinaten-liste⟩ 344 | ⟨Koordinatenliste⟩ 344 ; |

⟨Koordinatenliste⟩ 344 ,

⟨Koordinaten-liste⟩ 344 343 : = ⟨Koordinaten-paar⟩ 345 | ⟨Koordinaten-paar⟩ 345 ; ⟨Koordinaten-liste⟩ 344 |

⟨Koordinatenpaar⟩ 345 , ⟨Koordinatenliste⟩ 344

⟨Koordinaten-paar⟩ 345 344 : = (⟨Arithmetischer Ausdruck⟩ 49 , ⟨Arithmetischer Ausdruck⟩ 49) |

⟨Arithmetischer Ausdruck⟩ 49 , ⟨Arithmetischer Ausdruck⟩ 49

Die erste Version zur Darstellung der Koordinatenpaare war nach ANSI BASIC 82 ursprünglich vorgesehen, wurde aber im Rahmen der ANSI BASIC 82 Comments im Sinne der zweiten Version geändert.[277] Diese Version entspricht auch ANSI BASIC 79. Danach wird ein Koordinatenpaar nicht in Klammern eingeschlossen; dafür ist zur Trennung der Koordinatenpaare innerhalb der Koordinatenliste nur das Semikolon erlaubt.[278] Bei ANSI BASIC 79 kann die Koordinatenliste zudem mit einem Semikolon beginnen, was den Bildpunkt auf "ON" schaltet; diese Option ist nach ANSI BASIC 82 nicht mehr vorgesehen.
Ein einfaches PLOT-Statement ohne Plotliste schaltet den Bildpunkt aus (OFF), falls dies nicht bereits geschehen ist.[279]

276 ANSI BASIC 82, p. 163–165.
277 Vgl. ANSI BASIC 82 Comments, p. 93–96.
278 ANSI BASIC 79, p. 111.
279 ANSI BASIC 79, p. 113: "An empty plot-list shall cause the beam to be turned off and shall not change its current position." – ANSI BASIC 82, p. 165.

Ist der Bildpunkt ausgeschaltet (OFF), so wird er durch die Anweisung

```
PLOT X,Y        ! PLOT (X,Y)
```

ohne jede graphische Ausgabe zu den Koordinatenwerten (x, y) bewegt, dort ein- und anschliessend wieder ausgeschaltet. Damit wird die Ausgabe eines Punktes bewirkt.[280]

Bei der Anweisung

```
PLOT X,Y ;      ! PLOT (X,Y),
                ;
```

wird durch das abschliessende Semikolon bewirkt, dass der Bildpunkt nach Erreichung der Position (x, y) eingeschaltet (ON) bleibt.[281]
Bei der nach ANSI BASIC 79 möglichen Anweisung

```
PLOT ; X,Y
```

bewirkt das vorangehende Semikolon, dass der Bildpunkt zunächst eingeschaltet und dann in die neue Position (x, y) gebracht wird. Damit wird von der ursprünglichen Position eine Gerade bis zur neu spezifizierten Bildpunktposition gezogen; alsdann wird der Bildpunkt ausgeschaltet (OFF).
Nach ANSI BASIC 82 ist dieser Zeichenvorgang nur durch eine Kombination von PLOT- und SET BEAM-Statement möglich.[282]

```
SET BEAM "ON"
PLOT (X,Y)
```

Durch die Anweisung in ANSI BASIC 79-Norm

```
PLOT ; X,Y ;
```

bzw. nach ANSI BASIC 82 durch die Statementkombination

```
SET BEAM "ON"
PLOT (X,Y),
```

280 ANSI BASIC 79, p. 113: "If the beam is initially off, execution of a plot-statement with a plot-list consisting of a single coordinate-pair and no semicolon shall result in a dot being drawn at that point and the beam being turned back off." – ANSI BASIC 82, p. 165.
281 ANSI BASIC 79, p. 113: "A plot-list consisting of a single semicolon shall cause the beam to be turned on at the current position."
282 Zum SET BEAM-Statement vgl. Abs. 334.315.

wird erreicht, dass der Bildpunkt auch nach Erreichung der neuen Position eingeschaltet bleibt.[283]
Falls in einem PLOT-Statement mehrere Koordinatenpaare – jeweils getrennt durch ein Semikolon oder Komma – erscheinen, erfolgt die graphische Ausgabe der entsprechenden Linienzüge. Dies ist auch dann der Fall, wenn sich mehrere PLOT-Statements logisch folgen und jeweils mit einem Semikolon bzw. Komma abgeschlossen werden.[284] Als Massstabsdimension wird beim PLOT- sowie beim (nachfolgend erläuterten) MAT PLOT- und MAT FILL-Statement grundsätzlich die Problemeinheit (PE) gewählt.

334.312 MAT PLOT-Statement

Ebenso wie für das elementare PRINT-Statement[285] durch das MAT PRINT-Statement[286] eine Ausgabeanweisung zur Verarbeitung ganzer Felder entwickelt wurde, führt ANSI BASIC 82 das entsprechende Korelat zum PLOT-Statement in Form des MAT PLOT-Statements ein.[287] Die Statementstruktur besteht aus dem Kennwort MAT PLOT, einer eventuellen Begrenzung der zu zeichnenden Punktzahl und einer Felderausgabeliste.[288]

283 Vgl. hierzu auch Luehrmann, An Elaboration of Some Thoughts on a Graphical Syntax in BASIC, p. 9: "The PLOT statement can be comprehended ... in terms of several general rules of syntax, a fact which should help users. *Rule 1.* If any X-, Y coordinate data follows the word PLOT, the beam will go there and, if it is not already on, will be turned on at that point. *Rule 2.* Unless a PLOT statement ends in a semicolon, the beam will be turned off after the statement is executed. *Rule 3.* If a PLOT statement ends in a semicolon, the beam will be left in whatever state it was last in."

284 ANSI BASIC 82, p. 165. – Vgl. hierzu auch Luehrmann, An Elaboration of Some Thoughts on a Graphical Syntax in BASIC, p. 9: "The statement PLOT X1, Y1; X2, Y2; X3, Y3 [allows] to draw a connecting line from X1, Y1 to X2, Y2 to X3, Y3, finishing with the beam off. This generalization leads to two additional rules ... *Rule 4.* Internal semicolons in a PLOT statement serve to delimit coordinate data and also to leave the beam in its previous state. *Rule 5.* If a PLOT statement contains too many coordinate data items to be typed on one line, it can be truncated after any semicolon, with the remainder typed as an additional PLOT statement."

285 Vgl. Bd. 1, Abs. 332.21.

286 Vgl. Bd. 2, Abs. 331.3.

287 ANSI BASIC 82, p. 163.

288 ANSI BASIC 82, p. 163 und 165.

⟨MAT PLOT-Statement⟩ : =
346 341

MAT PLOT ⟨Felderausgabeliste⟩ |
347

MAT PLOT ⟨Punktzahl-begrenzung⟩ : ⟨Felderausgabeliste⟩
348 347

⟨Felderausgabeliste⟩ : = ⟨Feldname⟩ |
347 346 41

⟨Feldname⟩ , ⟨Felderausgabeliste⟩
41 347

⟨Punktzahlbegrenzung⟩ : = ⟨Arithmetischer Ausdruck⟩
348 346 49

Die Felderausgabeliste besteht aus einem oder mehreren, durch Kommas getrennten, Feldnamen. Die zeichenbaren Felder dürfen demnach nur numerischen Typs sein. Die Verarbeitungsweise der Felder sieht vor, dass bei Vektoren jeweils ein Vektorpaar anzugeben ist, wobei der erste Vektor die x-Koordinaten und der zweite die y-Koordinaten liefert.[289] Bei Matrizen ist vorgesehen, dass ihre zweite Dimension zum Zeitpunkt der Ausführung des MAT PLOT-Statements nur den Wert 2 haben sollte. Die erste Spalte enthält dann die x-Koordinaten und die zweite Spalte die y-Koordinaten.[290]
Wenn eine Punktzahlbegrenzung vorgegeben ist, soll die Anzahl der zu zeichnenden Koordinatenpunkte auf den ganzzahlig gerundeten Wert des arithmetischen Ausdrucks beschränkt werden.[291]

289 ANSI BASIC 82, p. 165: "If the graphic-array-list consists of 2 vectors, the x-coordinates of the points to plot shall come from the first vector and the y-coordinates from the second vector."
290 ANSI BASIC 82, p. 165: "If the graphic-array-list consists of an array of size 2 in the second dimension, then the x-coordinates shall come from the array column with the smaller subscript and the y-coordinates shall come from the array column with the larger subscript."
291 ANSI BASIC 82, p. 165: "If a size-select is present the index shall be evaluated by rounding to an integer to obtain the number of points to plot."

Dies bedeutet, dass durch die Anweisung

```
2330 MAT PLOT 23: X,Y
```

aus den beiden Vektoren X und Y jeweils 23 Werte zu Koordinatenpaaren zusammengefasst und gezeichnet werden. Fehlt die Punktzahlbegrenzung,

```
3000 MAT PLOT A,B
```

so gilt als Punktzahlbegrenzung die Dimensionsgrenze des ersten Vektors, d. h. in diesem Falle diejenige von A.
Die syntaktische Struktur des MAT PLOT-Statements weicht durch diese Festlegung von der bisher geübten Praxis der Ausgabelisten bei MAT PRINT-Statements ab.[292] Es ist zu erwarten, dass zukünftige Normen diesen Unterschied aufheben und eine Anpassung bringen, so dass dann für die Struktur des MAT PLOT-Statements gelten könnte:[293]

⟨MAT PLOT-Statement⟩ : =
346 341

MAT PLOT ⟨Numerische MAT-Ausgabeliste⟩
349

⟨Numerische MAT-Ausgabeliste⟩ : = ⟨Numerische Eintragung⟩ | ⟨Feldname⟩ |
349 346 216 41

⟨Feldname⟩ , ⟨Numerische MAT-Ausgabeliste⟩ |
41 349

⟨Numerische Eintragung⟩ , ⟨Numerische MAT-Ausgabeliste⟩
216 349

292 Vgl. Bd. 2, Abs. 331.3.

293 Da ein MAT PLOT-Statement nur numerische Felder verarbeiten kann, muss die sonst beim MAT PRINT-Statement mögliche MAT-Ausgabeliste auf einen numerischen Typ begrenzt werden; vgl. Bd. 2, Abs. 331.31.

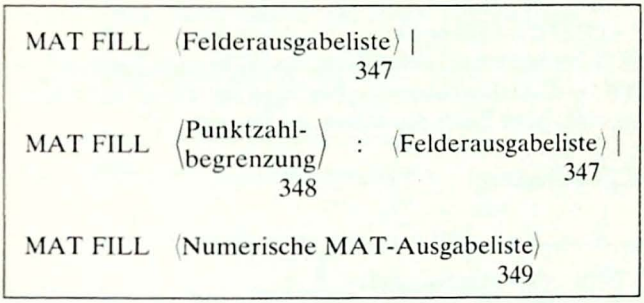
In dieser Form entfällt die Punktzahlbegrenzung und wird wie bei den MAT-Eingabe- und Ausgabelisten auf die Dimensionsgrenzen oder den vorhandenen Index-Trimmer[294] begrenzt.
Für das erste Beispiel des MAT PLOT-Statements müsste diesfalls die Anweisungen

```
2330 MAT PLOT X(23), Y(23)
```

geschrieben werden.

334.313 MAT FILL-Statement
Das MAT FILL-Statement unterscheidet sich strukturell vom MAT PLOT-Statement zunächst nur durch das Kennwort.[295]

⟨MAT FILL-Statement⟩ : =
350 341



Von ANSI BASIC 82 sind die beiden ersten Formen genormt. Es ist aber – wie beim MAT PLOT-Statement – zu erwarten, dass in die zukünftigen Normen die den Matrizenanweisungen angepasst dritte Form des MAT FILL-Statements Eingang finden wird.[296]
Die Wirkungsweise des MAT FILL-Statements entspricht insofern vollständig der des MAT PLOT-Statements, als anhand der Felderausgabeliste zunächst ein Polygonzug gezeichnet wird. Allerdings wird zusätzlich eine Verbindungslinie zwischen dem letzten und ersten Koordinaten-

294 Vgl. Bd. 2, S. 59–60, 112.
295 ANSI BASIC 82, p. 163.
296 Vgl. Abs. 334.312.

punkt gezogen.[297] Das von dem Polygonzug umschlossene Zeichenfeld wird anschliessend mit der im SET COLOR-Statement[298] angegebenen Farbe aufgefüllt.[299]
Durch die Anweisungsfolge

```
370 SET COLOR 4
380 MAT Y=ZERO(4)
390 MAT A=ZERO(4)
400 A(1)=0
410 A(2)=0
420 A(3)=5
430 A(4)=5
440 Y(1)=0
450 Y(2)=8
460 Y(3)=8
470 Y(4)=0
480 MAT FILL A, Y
```

wird ein farbiges Rechteck mit den Koordinatenpunkten (0,0;0,8;5,8;5,0) in der Farbe 4 gekennzeichnet und mit derselben Farbart angefüllt.

334.314 SET POSITION-Statement
Das SET POSITION-Statement dient dazu, den Bildpunkt zunächst auszuschalten (OFF) und ihn dann ohne graphische Ausgabe zu der im Koordinatenpaar spezifizierten Zeichenposition zu bewegen.[300]

⟨SET POSITION-Statement⟩ : =
351 341

SET POSITION ⟨Koordinatenpaar⟩ 345

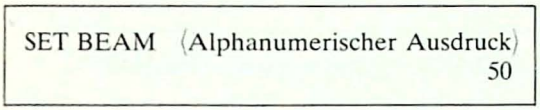
Im SET POSITION-Statement wird nur ein Koordinatenpaar angegeben; es wird in PE (Problemeinheiten) spezifiziert.

297 ANSI BASIC 82, p. 165: "The edges of the polygon consist of the sequence of line segments joining the points in order from the first trough the last and an additional line segment from the last point to the first."
298 Vgl. Abs. 334.211.
299 ANSI BASIC 82, p. 165: "The interior and edges of the polygon shall be filled in the current color as defined by SET COLOR, if supported by the graphics device."
300 ANSI BASIC 79, p. 113: "Execution of a set-statement with the keyword POSITION shall first turn the beam off, then move it to the point with the speci-

334.315 SET BEAM-Statement

Durch das SET BEAM-Statement kann der Bildpunkt an "ON" und ab "OFF" geschaltet werden.[301] Die Struktur ist einfach:[302]

⟨SET BEAM-Statement⟩ : =
352 341

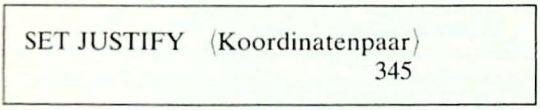


Die zulässigen Werte des alphanumerischen Ausdrucks dürfen nur "ON" oder "OFF" ergeben.

334.316 SET JUSTIFY-Statement

Das SET JUSTIFY-Statement[303] dient zur Positionierung der graphischen Ausgabe eines GRAPHIC PRINT-Statements.[304]

⟨SET JUSTIFY-Statement⟩ : =
353 341



Die Textausgabe des GRAPHIC PRINT-Statements soll so aufbereitet werden, dass die Werte des Koordinatenpaares relativ zum Rechteck, in dem die Textausgabe stehen wird, mit der gegenwärtigen Position des Zeichenpunktes übereinstimmen.[305]

fied problem coordinates, and finally leave the beam off." – ANSI BASIC 82, p. 164.

301 ANSI BASIC 82, p. 166: "Execution of a set-statement with the keyword BEAM shall turn the beam on if the value of the string-expression is 'ON' and shall turn the beam off if the value of the string-expression is 'OFF'. These values may be in any combination of upper- or lower-case-letters."

302 ANSI BASIC 82, p. 163.

303 ANSI BASIC 82, p. 163 und 166.

304 Vgl. Abs. 334.319.

305 ANSI BASIC 82, p. 166–167: "The label shall be displayed on the screen so that the point with the coordinates (h,v) relative to the rectangle coincides with the current position of the beam."

334.317 SET CHARACTER HEIGHT-Statement

Mit dem SET CHARACTER HEIGHT-Statement wird die Zeichenhöhe des Textes eines folgenden GRAPHIC PRINT-Statements in Problemkoordinaten festgelegt.[306]

⟨SET CHARACTER HEIGHT-Statement⟩ : =
354 341

SET CHARACTER HEIGHT ⟨Arithmetischer Ausdruck⟩ 49

334.318 SET TEXT ANGLE-Statement

Der arithmetische Ausdruck in einem SET TEXT ANGLE-Statement gibt den Winkel im Bogenmass in Uhrzeigerrichtung an, um welchen die Textausgabe eines GRAPHIC PRINT-Statements gegenüber der Horizontalen gedreht werden soll.[307] Für die Struktur gilt die einfache Festlegung:[308]

⟨SET TEXT ANGLE-Statement⟩ : =
355 341

SET TEXT ANGLE ⟨Arithmetischer Ausdruck⟩ 49

Nach ANSI BASIC 82 sind standardmässig Zeichensätze für 45, 90 und 180 Grad vorgesehen. Andere Winkelwerte sollen maschinenabhängig Ausgaben erzeugen.[309]

306 ANSI BASIC 82, p. 163 und 167: "Execution of a set-statement with the keywords CHARACTER HEIGHT sets the approximate height, in problem coordinates, of characters printed by subsequent graphic-print-statements."

307 ANSI BASIC 82, p. 167: "Execution of a set-statement with the keywords TEXT ANGLE sets the approximate direction of strings of characters printed by subsequent graphic-print-statements. The numeric-expression in such a set-statement represents the angle, in degrees or radians according to the current angle option, between the screen horizontal and a line through the approximate center of the characters in a counterclockwise direction."

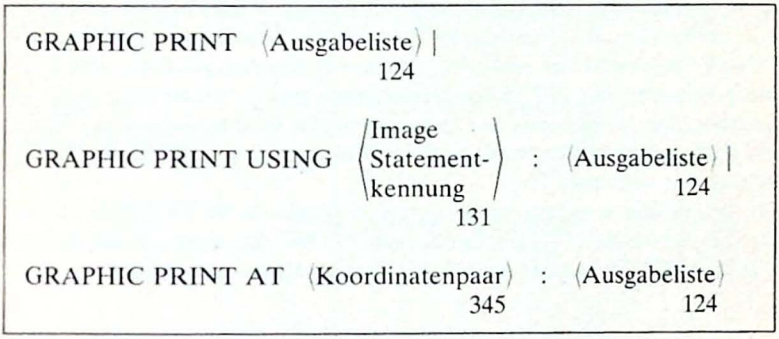
308 ANSI BASIC 82, p. 163.

309 ANSI BASIC 82, p. 167: "Implementation must display strings of characters horizontally for angles that are integer multiples of 180 degrees and vertically for angles that are odd multiples of 90 degrees. Text for angles that are odd multiples of 45 degrees shall be neither horizontally nor vertically oriented.

334.319 GRAPHIC PRINT-Statement

Zur graphischen Ausgabe von Zeichen dient das GRAPHIC PRINT-Statement. Seine Struktur unterscheidet sich bei ANSI BASIC 79 nur durch das Kennwort (GPRINT) von einem elementaren PRINT-Statement.[310] Mit ANSI BASIC 82 kann ausserdem noch die Image-Ausgabeliste[311] und überflüssigerweise eine Koordinatenangabe auftreten, die die sonst nahezu vollständige Äquivalenz zum PRINT- bzw. PRINT USING-Statement stört.[312]

⟨GRAPHIC PRINT-Statement⟩ : =
356 341



Nach der Syntax von ANSI BASIC 79 ist nur die erste Form möglich. Die graphische Ausgabeliste kann auch TAB-Funktionen[313] enthalten, deren Wirkungsweise im Rahmen von GPRINT-Statements allerdings nicht erläutert wird.

Einzelheiten bezüglich der zu generierenden Zeichenart und Zeichengrösse sowie der Zeichenabstände werden von ANSI BASIC (noch) nicht geregelt; sie können implementationsbezogen festgelegt werden.[314]

The orientation of text displayed at other angles shall be implementation-defined."

310 ANSI BASIC 79, p. 112.

311 Vgl. Bd. 1, S. 185–188.

312 ANSI BASIC 82, p. 163.

313 Zur TAB-Funktion vgl. Bd. 1, Abs. 332.22.

314 ANSI BASIC 79, p. 113: "The gprint-statement produces alphanumeric labels. The print-list shall be evaluated and the string of characters generated used as the label. The characters used for the labels shall have an implementation-defined size, style, and orientation; likewise, the spaces between characters and lines of text in a label shall be implementation-defined." – ANSI BASIC 82, p. 166.

Dagegen wird von ANSI BASIC 79 die zu realisierende Zeilenanordnung geregelt. Der Zeilenabstand ist generell nicht variierbar; die Randeinstellung kann vorgängig durch entsprechende PLOT- oder SET POSITION-Statements festgelegt werden.[315]
Endet eine graphische Ausgabeliste mit einem Semikolon, so verbleibt der Bildpunkt nach Abarbeitung der Ausgabeliste in der erreichten Position; andernfalls erfolgt ein Zeilenvorschub.[316]
Nach ANSI BASIC 82 kann durch das SET JUSTIFY-[317], SET CHARACTER HEIGHT-[318] und das SET TEXT ANGLE-Statement[319] die Textausgabeform gezielter beeinflusst werden. Keinerlei Aussagen finden sich aber darüber, wie die in der Ausgabeliste möglichen Trennzeichen Komma und Semikolon vom GRAPHIC PRINT-Statement interpretiert werden. Die Position des Bildpunktes wird durch das GRAPHIC PRINT-Statement nur verändert, wenn vor der Ausgabeliste eine Koordinatenangabe mit AT (Koordinatenpaar) steht. Andernfalls steht die Textausgabe linksbündig zur gegenwärtigen Bildpunktposition,[320] und die Bildpunktposition bleibt nach Ausführung des GRAPHIC PRINT-Statements erhalten.[321]
Die graphische Ausgabe kann weiterhin durch das SET CLIP-Statement beeinflusst werden.[322] Falls der Schalter "ON" ist, wird nur die via SET VIEWPORT-Statement festgelegte Zeichenfläche zur graphischen Aus-

315 ANSI BASIC 79, p. 113: "A newline generated in the course of evaluating a print-list shall cause the beam to be positioned at the current left margin one line space below its current position. The current left margin shall be that determined by the last x-coordinate established by default, by a plot-statement, or by a set-statement with the keyword POSITION; it shall not be affected by execution of gprint-statements."

316 ANSI BASIC 79, p. 113: "If the print-list ends with a semicolon, then the current position of the beam at the end of execution of a gprint-statement shall be the position in which it was left by the last character output; otherwise, the result shall be the same as if the last character of the label were a newline."

317 Vgl. Abs. 334.316.

318 Vgl. Abs. 334.317.

319 Vgl. Abs. 334.318.

320 ANSI BASIC 82, p. 167: "For example if (h, v) [Koordinatenpaar im SET JUSTIFY-Statement] equals (.5, .5), then the label will be centered about the current position of the beam, and if it equals (0,0), then the label will be lower-left justified at the current beam position."

321 ANSI BASIC 82, p. 166: "Execution of a graphic-print-statement shall not change the current position or state of the beam."

322 Vgl. Abs. 334.212.

gabe von Zeichen benutzt; ist der Schalter "OFF", so entfällt diese Begrenzung.[323]

334.32 Graphische Abfrageanweisungen

Die graphische Abfrageanweisungen dienen dazu, die gegenwärtige Bildpunktposition und eventuell die mit den SET BEAM-, SET JUSTIFY-, SET CHARACTER HEIGHT- oder SET TEXT ANGLE-Statement[324] festgelegten Werte zu eruieren und in die entsprechenden Variablen der einzelnen ASK-Statements zu übertragen. Nach ANSI BASIC 82 existieren analog zu den SET-Statements folgende graphischen Abfrageanweisungen.[325]

⟨Graphische Abfrageanweisung⟩ 357 340 : = ⟨ASK POSITION-Statement⟩ 358 |

⟨ASK BEAM-Statement⟩ 359 |

⟨ASK JUSTIFY-Statement⟩ 360 |

⟨ASK CHARACTER HEIGHT-Statement⟩ 361 |

⟨ASK TEXT ANGLE-Statement⟩ 362

323 ANSI BASIC 79, p. 113–114: "If clipping is enabled, then only those points and portions of line segments which lie inside the viewport shall be displayed. Likewise, if clipping is enabled, then characters which fall outside the viewport shall not be printed. In all cases, however, the current position of the beam shall be maintained, even though it is outside the viewport. If clipping is disabled, then all point, characters, and portions of line segments which lie inside the screen shall be displayed." – ANSI BASIC 82, p. 167–168.

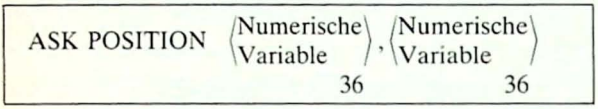
324 Vgl. Abs. 324.314 bis 334.318.

325 ANSI BASIC 82, p. 163–164.

In ANSI BASIC 79 ist nur das INQUIRE POSITION-Statement vorgesehen.

334.321 ASK POSITION-Statement
Das ASK POSITION-Statement wird von ANSI BASIC 79[326] und ANSI BASIC 82[327] in der gleichen Struktur genormt. Allerdings steht bei ANSI BASIC 79 anstelle des Kennwortes ASK das Kennwort INQUIRE.

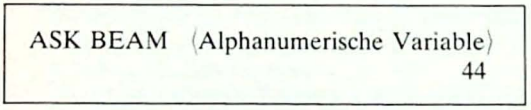
⟨ASK POSITION-Statement⟩ : =
358 357



Dabei übernimmt die erste Variable den x-Koordinatenwert, die zweite den y-Koordinatenwert. Beide Werte werden in PE (Problemeinheit) übertragen.[328]

344.322 ASK BEAM-Statement
Mit dem ASK BEAM-Statement wird der gegenwärtige Bildpunktzustand "ON" oder "OFF" in die alphanumerische Variable übertragen.[329]

⟨ASK BEAM-Statement⟩ : =
359 357



334.323 ASK JUSTIFY-Statement
Durch die Ausführung eines ASK JUSTIFY-Statements werden dem ent-

326 ANSI BASIC 82, p. 111.
327 ANSI BASIC 82, p. 163.
328 ANSI BASIC 79, p. 113: "Execution of an inquire-statement with the keyword POSITION shall assign the current position of the beam to the coordinate-variables in the inquire-statement." – ANSI BASIC 82, p. 166.
329 ANSI BASIC 82, p. 166: "Execution of an ask-statement with the keyword BEAM shall assign the string-variable the value 'ON' if the beam is on or the value 'OFF' if the beam is off."

haltenen numerischen Variablenpaar die gegenwärtigen Zentrierungswerte für eine graphische Ausgabe für ein GRAPHIC PRINT-Statement übergeben.[330]

⟨ASK JUSTIFY-Statement⟩ : =
360 357

ASK JUSTIFY ⟨Numerische Variable⟩ , ⟨Numerische Variable⟩
36 36

334.324 ASK CHARACTER HEIGHT-Statement
In die numerische Variable des ASK CHARACTER HEIGHT-Statements wird bei Ausführung der gegenwärtige Wert in Problemkoordinaten für die Zeichenhöhe der Zeichen eines GRAPHIC PRINT-Statements übertragen.[331]

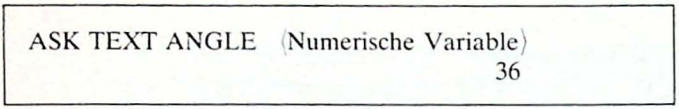
⟨ASK CHARACTER HEIGHT-Statement⟩ : =
361 357

ASK CHARACTER HEIGHT ⟨Numerische Variable⟩
36

334.325 ASK TEXT ANGLE-Statement
Mit dem ASK TEXT ANGLE-Statement kann der Neigungswinkel für eine graphische Textausgabe abgefragt werden.[332]

330 ANSI BASIC 82, p. 167: "Execution of an ask-statement with the keyword JUSTIFY shall assign the current centering values (h, v) to the coordinate-variables in the ask-statement."
331 ANSI BASIC 82, p. 167: "Execution of an ask-statement with the keyword CHARACTER HEIGHT shall assign to the numeric-variable the current value of character height."
332 ANSI BASIC 82, p. 167: "Execution of an ask-statement with the keywords TEXT ANGLE shall assign to the numeric-variable the current angle for text display in degrees or radians, depending on the angle option."

⟨ASK TEXT ANGLE-Statement⟩ : =
362 357



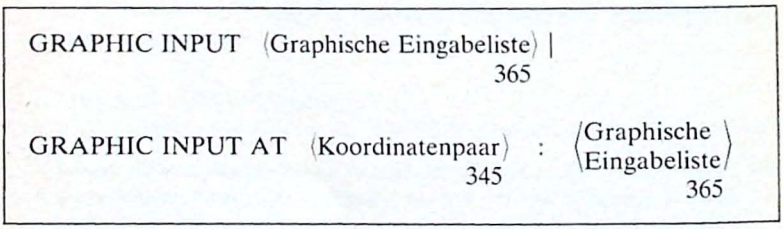
334.33 Graphische Eingabeanweisungen

Spezielle Bildschirmgeräte erlauben dem Benutzer die Direkteingabe von Daten mit Hilfe eines Lichtstiftes (light pen).[333] Diese Geräte können auch zur Direkteingabe von Koordinatenpunkten im Rahmen der Graphikanweisungen benutzt werden. Von ANSI BASIC wurde deshalb auch eine graphische Eingabeanweisung in Form des GRAPHIC INPUT-Statements in die Standards aufgenommen.[334] Es gilt somit:

⟨Graphische Eingabeanweisung⟩ : = ⟨GRAPHIC INPUT-Statement⟩
363 340 364

334.331 GRAPHIC INPUT-Statement
Für das GRAPHIC INPUT-Statement wird nach ANSI BASIC 82 folgende Struktur angegeben. Nach ANSI BASIC 79 ist nur die erste Form erlaubt und dann mit dem Kennwort GINPUT.[335]

⟨GRAPHIC INPUT-Statement⟩ : =
364 363



333 Vgl. Weber, Elektronische Datenverarbeitung, S. 37.
334 ANSI BASIC 79, p. 112. – ANSI BASIC 82, p. 163.
335 ANSI BASIC 79, p. 112.

Für die graphische Eingabeliste gilt:

⟨Graphische Eingabeliste⟩ 365 364 : = ⟨Numerische Variable⟩ 36 , ⟨Numerische Variable⟩ 36 |

⟨Numerische Variable⟩ 36 , ⟨Numerische Variable⟩ 36 , ⟨Alphanumerische Variable⟩ 44

Nach ANSI BASIC ist vom Benutzer zunächst ein Punkt mit dem Lichtstift auf dem Bildschirm anzugeben. Die Koordinatenwerte dieses Punktes werden in die numerischen Variablen übertragen.[336] Enthält die graphische Eingabeliste zusätzlich noch eine alphanumerische Variable, so werden in diese eventuell eingegebene Zeichen übertragen.[337] Über die Art der Stringeingabe fehlen bei ANSI BASIC nähere Angaben; es ist zu vermuten, dass sie nur über die Tastatur des Bildschirmgerätes (und nicht handschriftlich mit dem Lichtstift) erfolgen kann.
Durch die mit ANSI BASIC 82 mögliche Koordinatenangabe durch AT (Koordinatenpaar) wird der Bildpunkt vor Übertragung der Werte in die graphische Eingabeliste auf den angegebenen Koordinatenpunkt gesetzt.
Mit den bisher angeführten graphischen Anweisungen ist es möglich, alle denkbaren graphischen Bilder zu erzeugen. Das folgende BASIC-Programm basiert ausschliesslich auf den genormten ANSI BASIC-Statements. Seine Logik erzeugt abhängig von einer eingebbaren Zahl kreisförmige Polygonzüge. Das BASIC-Programm ist in der Norm von ANSI BASIC 79 geschrieben; die Änderungen nach ANSI BASIC 82 sind als Bemerkungen angefügt.

336 ANSI BASIC 79, p. 114: "The ginput-statement enables graphic input to be supplied during execution of a program. A gingput-statement is similar to an input-statement, except that instead of supplying actual values, the user must locate (e.g., by positioning cross-hairs) a point on the graphic display. The problem coordinates of that point shall be assigned to the coordinate-variables in the ginput-statement." – ANSI BASIC 82, p. 168.

337 ANSI BASIC 79, p. 114: "If a string-variable is present in the ginput-statement, then any characters supplied by the user to transmit the location of the point shall be assigned as the value of the string-variable." – ANSI BASIC 82, p. 168.

```
 5 REM STATEMENTS NACH ANSI BASIC 82 SIND DURCH EIN AUSRUFEZEICHEN (!)
 6 REM ERGAENZT.
 10 PRINT "PROGRAMM ZUR ERSTELLUNG VON KREISPOLYGONZUEGEN"
 20 SET BOUNDS 0,100, 0,135    ! SET BOUNDS 100/135, 1
 30 SET VIEWPORT 10,90, 10,90  ! SET VIEWPORT 0.1, 0.9, 0.1, 0.9
 40 REM MAXIMALER RADIUS DES POLYGONZUGES IN PROBLEMKOORDINATEN
 50 R = 100
 60 R1 = R * SQR(2)
 70 SET WINDOW -R1,R1, -R1,R1
 80 SET BRUSH 1                ! SET LINE STYLE 1  :  SET COLOR 1
 90 INPUT PROMT "ANZAHL DER POLYGONECKEN EINGEBEN ": N
100 FOR I=0 TO 360 STEP 360/N
110   B = DEG(I)
120   X = R * COS(B)
130   Y = R * SIN(B)
140   PLOT X,Y;               ! PLOT (X,Y),
150 NEXT I
160 INQUIRE POSITION X,Y  ! ASK POSITION X,Y
170 IF Y >= 0  GOTO 190
180 PLOT R,0;                 ! PLOT (R,0),
190 PLOT                      ! SET BEAM "OFF"
200 PRINT "NEUEN KREISPOLYGONZUG ZEICHNEN ?  JA ODER NEIN "
210 INPUT L$
220 IF L$ = "JA" GOTO 250
230 IF L$ = "NEIN" GOTO 320
240 GOTO 200
250 PRINT "BILDSCHIRM LOESCHEN ?      JA ODER NEIN "
260 INPUT L$
270 IF L$ = "JA" GOTO 300
280 IF L$ = "NEIN" GOTO 90
290 GOTO 250
300 CLEAR
310 GOTO 90
320 PRINT "ZEICHENPROGRAMM >POLYGONZUG< BEENDET"
330 END
```

Wird zum Beispiel als Anzahl der Polygonecken die Zahl 6 eingegeben, so entsteht Abbildung 17.

Abbildung 17.

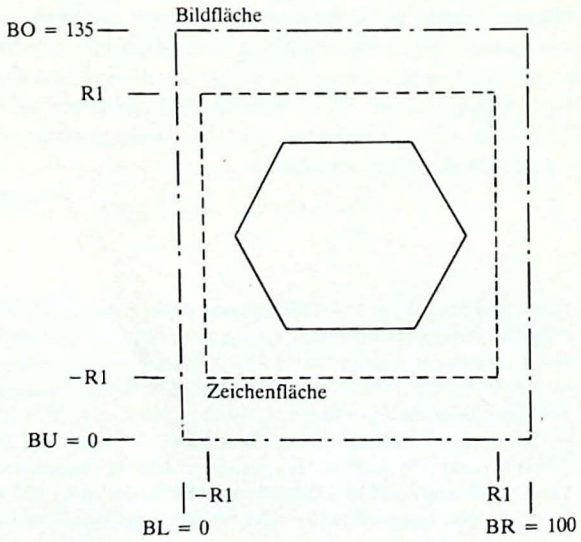


Abbildung 17.

Die gestrichelten Linien geben die Koordinatengrenzen des Bildschirms und des VIEWPORT resp. WINDOW an.
Es können während eines Programmablaufs auch mehrere Polygonzüge mit verschiedenen Eckenzahlen übereinander gezeichnet werden. Der Bildschirm darf hierzu vor der Eingabe der Eckenzahl für das nächste Polygon nicht gelöscht werden. Das Programm fragt diese Option ab. Werden nacheinander die Eckenzahlen 3, 4 und 6 eingegeben, so entsteht Abbildung 18.
Abbildung 18.

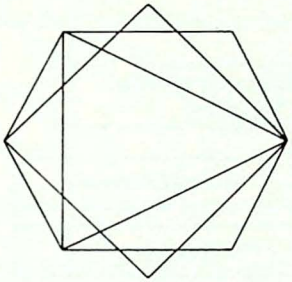


Abbildung 18.

Fehlermöglichkeiten, die bei Verwendung von Zeichenanweisungen bestehen, wurden bisher noch nicht umfassend spezifiziert. Fatale Fehler treten nach ANSI BASIC auf, wenn Zeichenanweisungen angewendet werden, obwohl die entsprechenden graphischen Geräte nicht verfügbar sind oder wenn ein GRAPHIC INPUT-Statement nicht vollständig ausgeführt werden kann.[338]

334.4 Bildunterprogrammanweisungen

Bei der Erstellung von technisch-wissenschaftlichen Zeichnungen ergibt sich für den Anwender immer wieder die Notwendigkeit, gleiche oder ähnliche Figuren – Kreise, Ellipsen, Rechtecke, Skalen, Geraden – innerhalb eines Programms mehrfach zu programmieren. Um diesen sich wie-

338 ANSI BASIC 79, p. 114: "Execution of a graphic output-statement occurs when there is no graphic output device available or specified. ... Execution of a graphic input statement occurs when there is no graphic input device available or specified. ... A ginput-statement containing a string-variable is executed and the user or input device fails to supply string input." – ANSI BASIC 82, p. 168–169.

derholenden Aufwand zu reduzieren, hat die ANSI BASIC-Norm in konsequenter Fortführung der Unterprogrammtechnik auch BASIC-Sprachanweisungen standardisiert, mit denen in sich selbständige Bildunterprogramme aufgebaut werden können.[339]
Für die Bildunterprogrammanweisungen gilt:

⟨Bildunterprogrammanweisung⟩ (366, 318) : = ⟨Call PLOT-Statement⟩ (367) | ⟨PICTURE-Block⟩ (372)

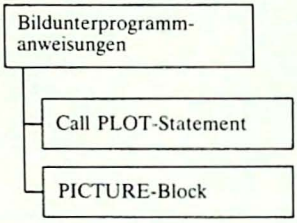


Abbildung 19.

334.41 Call PLOT-Statement

Analog zur Verarbeitung von Subroutinen[340] wird mit dem Call PLOT-Statement ein Bildprogramm in Form einer BASIC-Subroutine aufgerufen und der Programmablauf an dieses übergeben.
Nach ANSI BASIC kann mittels eines Call PLOT-Statements neben einer Parameterübergabe[341] auch gleichzeitig eine Transformation aller im Bildprogramm erzeugten Bildkoordinatenwerte erfolgen.[342] Dementsprechend kann das Call PLOT-Statement in unterschiedlichen Strukturen realisiert werden.

339 ANSI BASIC 79, p. 115: "Graphic procedures for drawing pictures can be defined in a manner similar to that for subprograms." – ANSI BASIC 82, p. 170.
340 Vgl. Bd. 2, Abs. 333.111.
341 Vgl. Bd. 2, Gleichung 298, S. 170 (CALL-Statement).
342 ANSI BASIC 79, p. 115: "Such picture-definitions can be invoked by a plot-statement, and the graphic output produced by them can be scaled, translated, and rotated by means of transformations specified in the plot-statement." – ANSI BASIC 82, p. 170.

⟨Call PLOT-Statement⟩ : =
367 366

PLOT ⟨Subroutinename⟩ |
298

PLOT ⟨Subroutinename⟩ (⟨Aktualparameterliste⟩) |
298 300

PLOT ⟨Subroutinename⟩ (⟨Aktualparameterliste⟩)
298 300

WITH ⟨Transformations-ausdruck⟩
368

Vermerkt sei, dass nach ANSI BASIC für das Call PLOT-Statement lediglich das Kennwort PLOT vorgesehen ist; dadurch können leicht Missverständnisse mit dem bereits erläuterten PLOT-Statement[343] auftreten; eindeutiger wäre die Verwendung der Kennwortkombination CALL PLOT.
Für den Unterprogrammnamen gelten die bereits bei Behandlung des CALL-Statements festgelegten Regeln.[344] Nach ANSI BASIC 79 kann der Unterprogrammname maximal 6 Zeichen umfassen.[345] Mit ANSI BASIC 82 wird dies auf 31 erhöht, wobei das erste Zeichen weiterhin ein Buchstabe sein muss.[346] Auch die Aktualparameterliste unterliegt den bereits besprochenen Regeln.[347] Mithin stimmen die beiden ersten Formen des Call PLOT-Statements bis auf das Kennwort PLOT vollständig mit dem CALL-Statement überein.
Die dritte Version des Call PLOT-Statements lässt die mit dem Kennwort WITH eingeleitete Verwendung eines Transformationsausdruckes zu.

343 Vgl. Abs. 334.311.
344 Vgl. Bd. 2, Abs. 333.11.
345 ANSI BASIC 79, p. 115: "Picture-names shall contain from one to six characters and shall be distinct from the names of variables, functions, pictures, and parallel-sections. A given picture-name shall not occur in more than one picture-statement in a program. A picture-name which occurs in a plot-statement shall also occur in a picture-statement."
346 ANSI BASIC 82, p. 170.
347 Vgl. Bd. 2, Abs. 333.11.

Dieses Statement ist dann zu benutzen, wenn eine Koordinatentransformation aller im PICTURE-Block erzeugten Koordinatenwerte erfolgen soll.
Der Transformationsausdruck kann aus einem oder mehreren, durch Sternzeichen (*) miteinander verbundenen, Transformationsangaben zusammengesetzt sein.[348]

⟨Transformations-ausdruck⟩ 368 : = ⟨Transformations-angabe⟩ 369 |

⟨Transformations-ausdruck⟩ 368 * ⟨Transformations-angabe⟩ 369

Transformationsangben sind in der Form

⟨Transformations-angabe⟩ 369 : = ⟨Feldname⟩ 40 | ⟨Transformations-funktion⟩ 370

möglich.

Der Aufbau der Transformationsfunktion unterliegt folgenden Strukturregeln:[349]

⟨Transformations-funktion⟩ 370 : = ⟨Transformations-funktionsname⟩ 371 (⟨Arithmetischer Ausdruck⟩ 49) |

⟨Transformations-funktionsname⟩ 371 (⟨Arithmetischer Ausdruck⟩ 49 , ⟨Arithmetischer Ausdruck⟩ 49)

Als Transformationsfunktionsnamen sind von ANSI BASIC bisher normiert:[350]

348 ANSI BASIC 79, p. 115. – ANSI BASIC 82, p. 172–173.
349 ANSI BASIC 79, p. 115. – ANSI BASIC 82, p. 170.
350 ANSI BASIC 79, p. 115. – ANSI BASIC 82, p. 170.

⟨Transformations-funktionsname⟩ : = ROTATE | SCALE | SHIFT | SHEAR | §§§

371 370

Die Zeichenfolge §§§ steht für compilerspezifische und zukünftig in die Standards aufzunehmende Transformationsfunktionsnamen.
Tabelle 3 zeigt die nach ANSIC BASIC vorgesehenen Transformationsfunktionen und ihre Wirkungsweise.[351]

Tabelle 3. Transformationsfunktionen

Funktion	Bedeutung
ROTATE (A)	Rotation des Koordinatenpaares (x,y) um A Grad im Uhrzeigersinn um den Nullpunkt des Problemkoordinatensystems. $(x,y) \rightarrow (\sqrt{x^2+y^2} \cdot \sin(\text{arctg}(\frac{y}{x}) - A),$ $\sqrt{x^2+y^2} \cdot \cos(\text{arctg}(\frac{y}{x}) - A))$
SCALE (A)	Massstabsvergrösserung des Koordinatenpaares (x,y) um den Faktor A. $(x,y) \rightarrow (Ax, Ay)$
SCALE (A, B)	Massstabsvergrösserung der x-Koordinate um den Faktor A und der y-Koordinate um den Faktor B. $(x,y) \rightarrow (Ax, By)$
SHIFT (A)	Verschiebung der Koordinatenwerte um den Wert A. $(x,y) \rightarrow (x+A, y+A)$
SHIFT (A, B)	Verschiebung des x-Koordinatenwertes um A und des y-Koordinatenwertes um B. $(x,y) \rightarrow (x+A, y+B)$
SHEAR (A)	Schneiden vertikaler Linien in die x-, y-Ebene, dabei um die x-Koordinate um A Grad geneigt. $(x,y) \rightarrow (x+y*\tan(A), y)$

351 ANSI BASIC 79, p. 117. – ANSI BASIC 82, S. 172. – Lediglich die Angabe SHIFT (A) fehlt in der Liste von ANSI BASIC; es ist jedoch zu vermuten, dass diese Funktion vergessen wurde, da die analoge Angabe bei der SCALE-Funktion möglich ist. – Vgl. auch: ANSI BASIC 82 Comments, p. 90–92.

Eine Transformation der Koordinatenwerte kann auch dadurch erreicht werden, dass mit der Transformationsliste in einer 4×4-Matrix ein beliebiges Transformationsfeld übergeben wird. Diesfalls wird der Koordinatenzeilenvektor

$$A = (c,cx,cx,0)$$

durch Matrizenmultiplikation

$$A \times G = N,$$

mit der Transformationsmatrix G, in den neuen Koordinatenzeilenvektor N umrechnet.[352]
Jede Transformation erzeugt zunächst eine spezielle Transformationsmatrix in der Dimension 4 * 4. Werden durch Sternzeichen mehrere Transformationsfunktionen oder auch Transformationsmatrizen miteinander verknüpft, so wird durch Matrizenmultiplikation des Transformationsausdrucks von links nach rechts die endgültige Transformationsmatrix und anschliessend der neue Koordinatenzeilenvektor erzeugt.[353]
ANSI BASIC legt für die Transformationsfunktion auch ein spezielles Transformationsstatement fest.[354] Es entspricht in seiner Struktur einem zusammengesetzten Matrizenstatement; es kann als graphisches MAT let-Statement[355] an beliebiger Stelle im Programm erscheinen und zur dynamischen Redimensionierung einer numerischen Matrix führen.[356]

352 ANSI BASIC 79, p. 116: "For the purpose of effecting transformations, the coordinates (x, y) of a point in two dimensions shall be represented by the 'homogeneous' coordinates (c, cx, cy, 0), where c is any nonzero number. Coordinates shall be transformed by postmultiplying this row vector by a 4×4 matrix, which is the value of the transform in the plot-statement, to obtain a new vector of homogeneous coordinates." – ANSI BASIC 82, p. 172.

353 ANSI BASIC 79, p. 116: "Particular transformations may be effected through the use of transform-functions, which have one or two numeric-expressions as arguments and which produce 4×4 matrices as their values." – ANSI BASIC 82, p. 172.

354 ANSI BASIC 79, p. 115. – ANSI BASIC 82, p. 170.

355 Vgl. Bd. 2, Abs. 331.223. – Das sogenannte „transform-assignment" wird im ANSI BASIC-Bericht nicht unter den MAT-Statements erwähnt; es erscheint nur in Verbindung mit den Graphikanweisungen.

356 ANSI BASIC 79, p. 117: "When a transform-assignment is executed, the 4×4 matrix, which is the value of the transform shall be assigned to the numeric-array to the left of the equals-sign, dynamically redimensioning that numeric-array if neccessary." – ANSI BASIC 82, p. 173.

Transformationsstatements im vorerwähnten Sinne machen es möglich, komplexe Call PLOT-Statements durch einfachere Formen zu ersetzen. So ist beispielsweise das Statement

```
2300 PLOT KREIS(R) WITH SHIFT(F,G) * SCALE(L)
```

durch

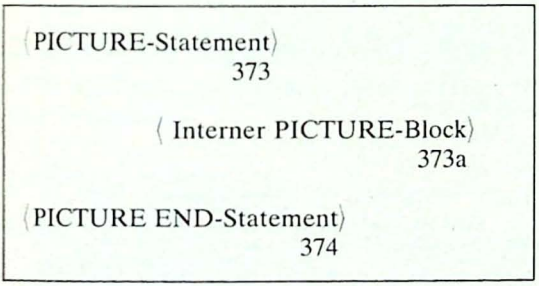
```
2300 MAT A = SHIFT(F,G) * SCALE(L)
2305 PLOT KREIS(R) WITH A
```

ersetzbar.

334.42 PICTURE-Block

Der PICTURE-Block stellt – analog zum SUB-Block[357] – ein eigenständiges Unterprogramm dar. Allerdings ist von ANSI BASIC noch kein PICTURE EXIT-Statement als Korrelat zum SUBEXIT-Statement vorgesehen.
Für die Struktur des PICTURE-Blocks gilt:

⟨PICTURE-Block⟩ : =
372 366



Das PICTURE-Statement ist bis auf das Kennwort strukturgleich mit dem SUB-Statement.[358]

357 Vgl. Bd. 2, Abs. 331.12.
358 ANSI BASIC 79, p. 115. – Vgl. Bd. 2, Abs. 333.12.

⟨PICTURE-Statement⟩ : =
373 372

PICTURE ⟨Subroutinename⟩ | 298
PICTURE ⟨Subroutinename⟩ (⟨Formalparameterliste⟩) 298 310

Für den Unterprogrammnamen und eine eventuelle Formalparameterliste gelten dieselben Regeln wie beim SUB-Statement.[359]
In ANSI BASIC 82 fehlt im internen PICTURE-Block zunächst ein zum SUBEXIT-Statement[360] analoges PICTURE EXIT-Statement. Es war übersehen worden und wird im zukünftigen Standard enthalten sein[361]; demzufolge gilt für den internen PICTURE-Block:

⟨Interner PICTURE-Block⟩ : = ⟨PICTURE EXIT-Statement⟩ \|
373a 372 373b

⟨Statement-Block⟩ \|

⟨Statement-Block⟩ ⟨Interner PICTURE-Block⟩
373a

Das PICTURE EXIT-Statement wird nur ein Kennwort enthalten.

⟨PICTURE EXIT-Statement⟩ : =
373b 373a

EXIT PICTURE

Die Liste der im Statement-Block zugelassenen BASIC-Statements ist um einige Statements vermindert. So dürfen keine Koordinatenanweisungen, keine Bildpunktanweisungen, keine Anweisungen des BASIC-

359 Vgl. Bd. 2, Abs. 333.12.
360 Vgl. Abs. 333.12.
361 ANSI BASIC 82 Comments, p. 100: "This was an oversight and has been added to the draft editorially."

Unterprogramms (ausser dem CALL-Statement) und keine END-Statements vorkommen.[362]
Für das PICTURE END-Statement gilt:[363]

⟨PICTURE END-Statement⟩ : =
374 372

END PICTURE

Es darf nur einmal in einem PICTURE-Block und dann nur am Ende der Statementfolge auftreten.
Rotiert man ein Quadrat mit der Seitenlänge A um seinen Schwerpunkt in Winkelschritten von R Grad, wobei nach jedem Rotationsschritt die Kantenlänge um den Faktor (SIN(R) + COS(R)) erhöht wird, so ergibt sich bei 10 Rotationsschritten zu jeweils 10 Grad die Figur in Abbildung 20.
Unter der Voraussetzung, dass ein entsprechendes graphisches Terminal oder ein Plotter vorhanden ist und der verwendete BASIC-Prozessor vollständig innerhalb der ANSI BASIC-Norm liegt, kann mit dem folgenden BASIC-Programm die Abbildung 20 erstellt werden.
Die Zeichnung des Vierecks wird dabei jeweils im PICTURE-Block mit dem Namen VIERECK durchgeführt. Beim Aufruf des Bildprogramms VIERECK wird durch die Transformationsfunktion SCALE(S) die Vergrösserung der Kantenlänge und durch ROTATE (R1) die entsprechende Rotation um den Schwerpunkt erzeugt. Die Normanpassung nach ANSI BASIC 82 ist wiederum als Bemerkung im folgenden BASIC-Programm enthalten.

```
100 REM VIERECKROTATION
110 PRINT "AUSGANGSGROESSE DES VIERECKS IN PROZENT DER ZEICHENFLAECHE"
115 INPUT PROMT "EINGEBEN ": A
120 INPUT PROMT "ROTATIONSWINKEL DES VIERECKS EINGEBEN ": R
130 INPUT PROMT "ANZAHL DER ROTATIONSSCHRITTE EINGEBEN ": N
140 SET BOUNDS 0,100, 0,140          ! SET BOUNDS 5/7, 1
150 SET VIEWPORT 0,100, 20,120       ! SET VIEWPORT 0, 0.8, 1/7, 6/7
160 SET WINDOW -51,51, -51,51
```

362 ANSI BASIC 79, p. 116: "A picture-definition shall be executed in the same fashion as is a subprogram, with two exceptions. First, set-statements which redefine coordinate systems (i.e., those with keywords BOUNDS, VIEWPORT, and WINDOW) or which affect clipping (i.e., that with the keyword CLIP) shall not be executed during the execution of a picture-definition. And second, the coordinates of all points generated during execution of a picture-definition shall be transformed." – ANSI BASIC 82, p. 171.
363 ANSI BASIC 79, p. 115. – ANSI BASIC 82, p. 170.

```
170 SET BRUSH 1                    ! SET LINE STYLE 1 : SET COLOR 1
180 IF A > 100 GOTO 310
190 B = DEG(R)
195 R1 = 0
200 S = SIN(B)+COS(B)
210 PLOT VIERECK(A)
220 FOR I=1 TO N
230   PLOT VIERECK(A) WITH SCALE(S) * ROTATE(R1)
240   S = S*S
250   R1 = R1+R
255   IF S*A > 100 GOTO 290
260 NEXT I
270 PRINT "ZEICHNUNG BEENDET"
280 STOP
290 PRINT "ZEICHNUNG BEENDET, DA ZEICHENFLAECHE VOLLSTAENDIG AUSGENUTZT"
300 STOP
310 PRINT "ZEICHNUNG NICHT MOEGLICH, DA AUSGANGSVIERECK ZU GROSS"
320 END
1000 PICTURE VIERECK(A)
1010 LET B = A/2
1020 SET POSITION B,-B
1030 PLOT : B,B; -B,B; -B,-B; B,-B
1040 END PICTURE
```

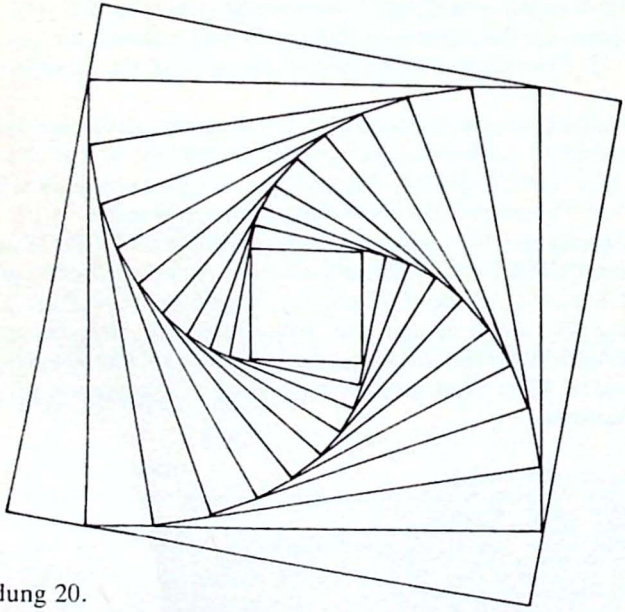


Abbildung 20.

335 Diagnoseanweisungen

Da beim Test oder bei der Übernahme umfangreicher BASIC-Programme die normale syntaktische, statementbezogene Fehlerkontrolle den Anwender nur wenig unterstützt, bieten die meisten BASIC-Programmiersysteme zusätzlich sogenannte Diagnoseanweisungen an. Sie

dienen insbesondere zum erleichterten Auffinden logischer Progammierfehler (debugging) und zur Ablaufkontrolle beim Programmtest.[364]
Nach ANSI BASIC werden alle Diagnoseanweisungen den BASIC-Sprachanweisungen zugeordnet, da sie ausschliesslich innerhalb von BASIC-Programmen vorkommen dürfen.[365]
Bei manchen BASIC-Programmiersystemen können jedoch einzelne Diagnoseanweisungen nur ausserhalb der BASIC-Programme – als BASIC-Systemanweisungen – auftreten. Aus diesem Grund werden die Diagnoseanweisungen in Abbildung 1 (S. 13) sowohl den BASIC-Sprachanweisungen als auch den BASIC-Systemanweisungen zugeordnet.
Unabhängig von der vorerwähnten Einteilung kann generell zwischen Systemdiagnoseanweisungen und Programmdiagnoseanweisungen differenziert werden. Vgl. Abbildung 21.

⟨Diagnose-anweisung⟩ : = ⟨Systemdiagnose-anweisung⟩ | ⟨Programmdiagnose-anweisung⟩
375 294 376 379

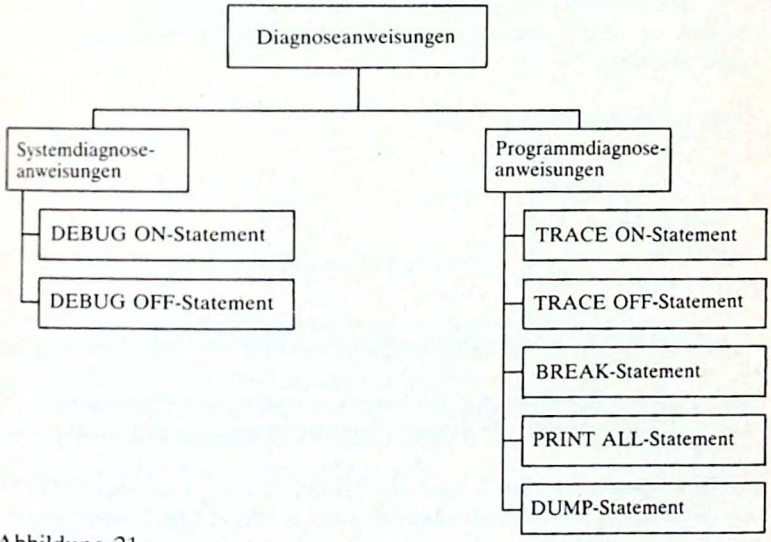


Abbildung 21.

364 Vgl. auch Bd. 1, Abs. 234. – ANSI BASIC 79, p. 135: "Debugging facilities are provided by language statements in order to allow test points to be built into a program. These statements allow the user to set break points, to trace the action of the program, and to turn the debugging system on and off within each program-unit." – ANSI BASIC 82, p. 154.
365 ANSI BASIC 79, p. 135–136. – ANSI BASIC 82, p. 154–155.

Die folgende Darstellung der Diagnoseanweisungen hält sich weitgehend an die ANSI BASIC-Norm; mitberücksichtigt werden auch einige FOSBIC-spezifische Diagnoseanweisungen.

335.1 Systemdiagnoseanweisungen

Die Systemdiagnoseanweisungen dienen zur Ein- und Ausschaltung des Diagnosesystems (debugging system).

⟨Systemdiagnose-anweisung⟩ 376 : = ⟨DEBUG ON-Statement⟩ 377 | ⟨DEBUG OFF-Statement⟩ 378

335.11 DEBUG ON-Statement

Durch das DEBUG ON-Statement werden das Diagnosesystem und eventuell im BASIC-Programm vorhandene Programmdiagnoseanweisungen aktiviert.[366]

⟨DEBUG ON-Statement⟩ 377 : = 376

```
DEBUG ON |
DEBUG
```

Nach ANSI BASIC ist nur die erste Statementform als Standard vorgesehen.
Ohne explizite Einschaltung des Diagnosesystems ist dieses inaktiv.[367] Eventuell vorhandene Programmdiagnoseanweisungen bleiben wirkungslos.
Bei der FOSBIC Version 2 wird das DEBUG ON- (und das DEBUG OFF-)Statement – in Übereinstimmung mit ANSI BASIC – den Sprachanweisungen zugeordnet. Demgegenüber werden die DEBUG-State-

366 ANSI BASIC 79, p. 135: "Execution of the debug-statement DEBUG ON shall cause debugging to become active in the program-unit in which the debug-statement occurs." – ANSI BASIC 82, p. 154.

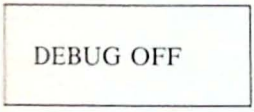
367 ANSI BASIC 79, p. 135: "Prior to the execution of any debug-statement in a program-unit, debugging shall be inactive." – ANSI BASIC 82, p. 154.

ments bei manchen anderen BASIC-Programmiersystemen als BASIC-Systemanweisungen behandelt. Teilweise entspricht die Wirkung der DEBUG-Statements auch nicht der ANSI BASIC-Norm.
Beim System Nixdorf 8870/1 wird durch die DEBUG-Systemanweisung eine schrittweise Programmbearbeitung eingeleitet, die nach jedem abgearbeiteten BASIC-Programmstatement einen Programmstop verursacht; die Programmfortführung muss vom Benutzer durch Betätigung der »CR-Taste«[368] nach jedem Statement ausgelöst werden. Grundsätzlich besteht auch die Möglichkeit, im DEBUG-Statement die Statementnummer zu spezifizieren, ab der eine Programmdiagnose erfolgen soll.[369]

3335.12 DEBUG OFF-Statement

Durch das DEBUG OFF-Statement wird das Diagnosesystem ausgeschaltet;[370] die im BASIC-Programm enthaltenen Programmdiagnoseanweisungen werden damit generell unwirksam.

(DEBUG OFF-Statement) : =
3778 376



3335.2 Programmdiagnoseanweisungen

Durch Programmdiagnoseanweisungen werden die zwischen einem logisch sich folgendem Paar von DEBUG ON- und DEBUG OFF-Statements liegenden BASIC-Sprachanweisungen analysiert.
Nach ANSI BASIC sind nur TRACE- und BREAK-Statements als Programmdiagnoseanweisungen vorgesehen[371]; unter Mitberücksichtigung FOSBIC-spezifischer Anweisungen gilt:

3668 Die „CR-Taste" löst die Übergabe einer eingegebenen Bildschirmzeile an den Rechner aus.
3669 Nixdorf, System-Software Business-Basic, S. 2.5.
3670 ANSI BASIC 79, p. 135: "Execution of the debug-statement DEBUG OFF shall cause debugging to become inactive in that program-unit." – ANSI BASIC 82, p. 154.
3671 ANSI BASIC 79, p. 135. – ANSI BASIC 82, p. 154.

⟨Programmdiagnose-anweisung⟩ 379 : = ⟨TRACE ON-Statement⟩ 376 | ⟨TRACE OFF-Statement⟩ 380 | ⟨BREAK-Statement⟩ 381 | ⟨PRINT ALL-Statement⟩ 382 | ⟨DUMP-Statement⟩ 383 384

335.21 TRACE ON-Statement

Das TRACE ON-Statement dient zur Verfolgung der Progammlogik (logical trace) und/oder zur Inhaltsangabe der Variablen, die während der Programmausführung durchlaufen werden (variable trace).[372]
Für die Statementstruktur gilt generell:

⟨TRACE ON-Statement⟩ 380 : = 379

TRACE ON |

TRACE |

TRACE ON AT # ⟨Arithmetischer Ausdruck⟩ 49 |

TRACE ⟨Variablenliste⟩

Nach ANSI BASIC 79 ist nur die erste Form vorgesehen;[373] die FOSBIC Version 2 lässt die beiden ersten Formen zu. ANSI BASIC 82 normiert die ersten drei Formen.[374]
Durch TRACE ON wird nach ANSI BASIC sowohl eine logische als auch

372 Vgl. Terry M. Walker, Introduction to Computer Science: BASIC Language Programming. Boston, p. 75–76: "There are two kinds of trace statements: (1) logical trace statement, and (2) variable trace statement. The objective in a logical trace is to determine how far program execution has proceeded and which branches are being made. ... In a variable trace the interest is in printing the values as they are assigned to each of the key variables in the program. This permits us to determine if intermediate processing is producing the correct results."
373 ANSI BASIC 79, p. 135.
374 ANSI BASIC 82, p. 154.

eine inhaltsmässige Programmkontrolle eingeleitet. Sie ist abhängig von der Statementart und soll bei den Programmsteuerungsanweisungen[375] die Statementnummer des gerade ausgeführten und des nächsten Programmstatements anzeigen. Anweisungen, die Variable enthalten, insbesondere die Eingabeanweisungen, sollen mit allen Variablenwerten und der Statementnummer aufgelistet werden.[376] Bei den Ausgabeanweisungen, ausgenommen den Fileausgabeanweisungen, genügt normalerweise die Angabe der Statementnummer.
Form und Format der durch das TRACE ON-Statement erzeugten Ausgabe können compilerspezifisch festgelegt werden.[377]
Ein TRACE ON-Statement kann nur zur Wirkung kommen, wenn das Diagnosesystem zuvor durch ein DEBUG ON-Statement aktiviert wurde.[378] Der vorerwähnte, nach ANSI BASIC vorgesehene, Wirkungszusammenhang besteht bei manchen BASIC-Compilersystemen allerdings nicht.
Das nachfolgende Programmbeispiel ist nach HB BASIC (Honeywell Bull) zulässig und entspricht auch vollständig der ANSI BASIC-Norm.

```
10 REM HB BASIC
20 READ X, Y, Z
30 TRACE ON
40 LET A = X + (Y * Z)
50 IF A = 0 THEN 90
60 PRINT " A NOT ZERO"
70 GOTO 20
80 DATA -9, 3, 4, -12, 3, 4
90 TRACE OFF
100 END


*AT 40
*AT 50
*AT 60
  A NOT ZERO
*AT 70
*AT 40
*AT 50
*AT 100
```

375 Zu den Programmsteuerungsanweisungen vgl. Bd. 1, Abs. 332.4.

376 ANSI BASIC 79, p. 135–136: "Whenever tracing is on and debugging is active in a program-unit, the following actions shall occur each time a statement of the specified type is executed: (a) for a control-statement, next-statement, end-loop-statement, or call-statement, both the line-number of the statement and the line-number of the next statement executed shall be reportet; and (b) for a let-statement, input-statement, linput-statement, read-statement, mat-input-statement, mat-linput-statement, mat-read-statement, or inquire-statement, both the line-number of the statement and the values assigned to any variables by the statement shall be reported." – ANSI BASIC 82, p. 154–155.

377 ANSI BASIC 79, p. 136: "The form of all trace reports is implementation-defined." – ANSI BASIC 82, p. 155.

378 ANSI BASIC 79, p. 135: "The execution of a trace-statement when debugging is inactive shall have no effect." – Vgl. Abs. 335.11.

Um die Wirkung des TRACE ON-Statements zu aktivieren, müsste nach ANSI BASIC vor dem Statement 30 noch die Anweisung

```
25 DEBUG ON
```

eingefügt werden.

Eine weitere Form der Ausgabemöglichkeit durch das TRACE ON-Statement, hier mit implizitem ON, zeigt das nächste Beispiel.[379]

```
  5 TRACE
 10 REM EINFACHES ZEICHENPROGRAMM
 20 FOR X = 3 TO 100 STEP 5
 30    LET Y = SIN(DEG(X))
 40    LET Z = COS(DEG(X))
 50    LET Y = 20 + (20*Y)
 60    LET Z = 20 + (20*Z)
 70    PRINT X;
 80    IF Z > Y THEN 110
 90    PRINT TAB(Z); "*"; TAB(Y); "+"
100    GOTO 120
110    PRINT TAB(Y); "+"; TAB(Z); "*"
120 NEXT X
130 END


AT 20, X=3
AT 30, Y=3
AT 40, Z=1
AT 50, Y=20
AT 60, Z=40
AT 70
AT 110
AT 120, NEXT AT 20
AT 20, X=5
 ...
 ...
```

Bei grösseren BASIC-Programmen kann das TRACE ON-Statement sehr rasch zu umfangreichen und unübersichtlichen Ausgabelisten führen; es ist deshalb nur sehr zurückhaltend zu verwenden.[380] Aus diesem Grunde besteht auch bei ANSI BASIC 82 die dritte Form des TRACE ON-Statements, mit dem die von dem TRACE ON-Statement erzeugte Ausgabe über die im arithmetischen Ausdruck angegebene Kanalnummer auf ein File übertragen wird.[381]

379 Eugene H. Barnett. Programming Time-Shared Computers in BASIC. New York 1972, p. 137.

380 Fred Gruenberger. Computing with the BASIC Language. New York 1972, p. 88. "As a debugging tool, tracing is to be regarded as a last-ditch resort, and should never be used casually. Using a tool crude as tracing for debugging is the mark of a poor programmer."

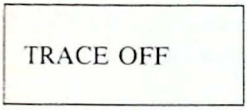
381 ANSI BASIC 82, p. 155: "Whenever tracing has beeen turned on via a trace-statement with a channel-expression, trace reports shall be directed to the (display format) file assigned to the specified channel."

Bei einzelnen BASIC-Programmiersystemen besteht die Möglichkeit, die Ausgabe von Variablenwerten nur auf diejenigen zu beschränken, die im TRACE ON-Statement in einer Variablenliste aufgeführt sind. Diese Statementform ist beispielsweise bei NEWBASIC zulässig.[382]
TRACE ON-Statements können in dem zu kontrollierenden Programm an beliebiger Stelle eingefügt werden.

335.22 TRACE OFF-Statement

Durch das TRACE OFF-Statement wird die mit TRACE ON initialisierte Programmanalyse aufgehoben.

⟨TRACE OFF-Statement⟩ : =
381 379

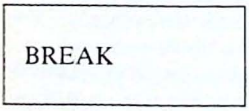


Das Statement wirkt nur bei aktiviertem Diagnosesystem (DEBUG ON).[383]
Soweit das DEBUG OFF-Statement als BASIC-Sprachanweisung verwendbar ist, kann es an die Stelle eines TRACE OFF-Statements treten; das Diagnosesystem wird diesfalls bei Erreichung des TRACE OFF-Statements allerdings gänzlich abgeschaltet.

335.23 BREAK-Statement

Das nach ANSI BASIC vorgesehene

⟨BREAK-Statement⟩ : =
382 379



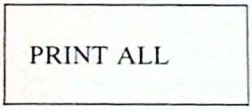
382 Com-Share Incorporated. NEWBASIC Reference Manual, Pittsburgh 1971, p. 133–134.
383 Vgl. hierzu Fussnote 378 und Abs. 335.11.

stellt die allgemeinste Form einer Diagnoseanweisung dar.[384] Sie kann – im Gegensatz zu den übrigen Programmierdiagnoseanweisungen – nur im Dialogverkehr benutzt werden.
Sobald die Programmausführung – bei aktivem Diagnosesystem – ein BREAK-Statement durchläuft, kommt es zu einem Programmstop. Gleichzeitig wird die Statementnummer des BREAK-Statements angegeben und dem Benutzer angezeigt, dass nunmehr eine Interaktionsmöglichkeit mit dem aktiven Diagnosesystem besteht.[385] Art und Form dieser Interaktionsmöglichkeit können compilerspezifisch festgelegt werden; entsprechendes gilt nach ANSI BASIC für die Form der Programmfortsetzung.[386] Praktisch dürfte es immer möglich sein, entweder das Diagnosesystem auszuschalten (DEBUG OFF), mit der Programmbearbeitung fortzufahren oder sie endgültig abzubrechen.
Bei inaktivem Diagnosesystem bleibt ein BREAK-Statement ohne Wirkung.[387]

335.24 PRINT ALL-Statement

Das PRINT ALL-Statement ist beim UWBIC- und FOSBIC-Compiler verfügbar. Es weist die Form

⟨PRINT ALL-Statement⟩ : =
383 379



auf. Durch dieses Statement werden die temporären Werte aller Variablen zeilenweise ausgedruckt, sofern sie einen festlegbaren kleinsten Absolutwert nicht unterschreiten. Ebenso wird eine Übersichtsliste aller bis

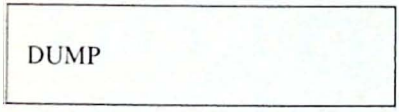
384 ANSI BASIC 79, p. 135.
385 ANSI BASIC 79, p. 135: "The execution of a break-statement when debugging is active shall cause an exception. The standard recovery procedure from this exception shall be to report the line-number of the break-statement and to signify to the user that interaction with the debugging system is possible." – ANSI BASIC 82, p. 154.
386 ANSI BASIC 79, p. 135: "The actions allowed by the debugging system, including the method for continuing execution of the program, are implementation-defined." – ANSI BASIC 82, p. 154.
387 ANSI BASIC 79, p. 135: "The execution of a break-statement when debugging is inactive shall have no effect." – ANSI BASIC 82, p. 155.

zum PRINT ALL-Statement angelegten Dateien gedruckt. PRINT ALL ist mit der dritten Form des TRACE ON-Statements vergleichbar,[388] wobei an Stelle einer Variablenliste die Kennung ALL auftritt.
Das PRINT ALL-Statement sollte generell nur mit grösster Zurückhaltung verwendet werden, da bei seiner Anwendung u. U. umfangreiche Drucklisten anfallen.[389]

335.25 DUMP-Statement

Das DUMP-Statement bewirkt die Ausgabe des Programms in compiler- bzw. maschinenspezifischer Form. Insofern ist es lediglich für die Systemprogrammierung bzw. bei der Compilerimplementation von Nutzen. Die Weiterentwicklung dieser Diagnoseanweisung zu einer benutzerfreundlichen Form stellen die in Abschnitt 45 erwähnten Cross-Reference-Listen dar.
Das DUMP-Statement setzt – beim UWBIC- und FOSBIC-Compiler – die Durchnumerierung aller BASIC-Statements voraus. Es weist die Struktur

⟨DUMP-Statement⟩ : =
384 379



auf und ist inbesondere zur Compilerimplementierung nützlich. Das Statement ist auch geeignet, interessierten Systemanalytikern Einblick in die Compilerstruktur zu vermitteln.[390]
Bei Betrachtung des folgenden Beispiels ist insbesondere auf die verwendete Ziffernverschlüsselung – beispielsweise für PRINT – und die Art der Speicherplatzreservierung zu achten.

388 Vgl. Abs. 335.21.
389 Eine genaue Erläuterung der via PRINT ALL-Statement erzeugten Druckausgaben findet sich bei Weber und Türschmann, FOSBIC-Compiler, Abs. 332.3.
390 Zur Struktur des FOSBIC-Compilers vgl. Karl Weber und Carl Wolfram Türschmann. FOSBIC-Compiler. Ein BASIC-Compilersystem auf FORTRAN-Basis. Bern und Stuttgart 1977, Abs. 32.

```
10 DUMP
15 REM PROGRAMM ZUR BERECHNUNG DES MITTELWERTES UND DER VARIANZ
20 READ A
30 IF A$="ENDE" GOTO 55
35 LET S1=S1+A
40 LET S2=S2+A**2
45 LET N=N+1
50 GOTO 20
55 PRINT "   ANZAHL","MITTELWERT","  VARIANZ"
60 PRINT N,S1/N,(N*S2-S1**2)/(N*(N-1))
65 STOP
70 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
75 DATA "ENDE"
80 END


 CONSTANTS

LOCATION        VALUE
     314             0.00000
     315             1.00000
     316             2.00000
     317             3.00000
     318             4.00000
     319             5.00000
     320             6.00000
     321             7.00000
     322             8.00000
     323             9.00000


  ANZAHL       MITTELWERT        VARIANZ
    17                9              25.5
```

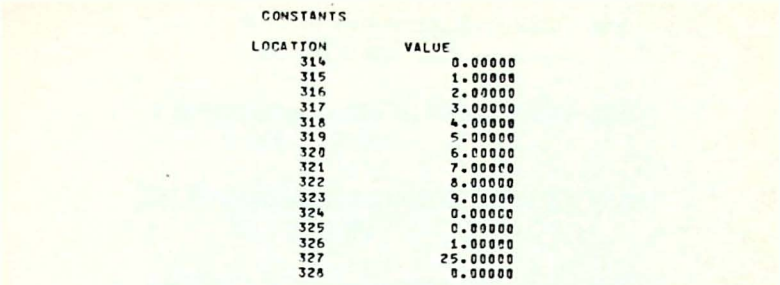
Der von dem Beispiel

```
10 DUMP
20 DIM A(25),B(10,15)
30 MAT READ A
40 MAT READ B
50 FOR I=1 TO 25
55   READ A(I)
60 NEXT I
65 FOR I=1 TO 10
70   FOR J=1 TO 15
75     READ B(I,J)
80   NEXT J
90 NEXT I
95 END
```

erzeugte DUMP-Ausdruck des FOSBIC-Codes zeigt die geringe Speicherplatzbeanspruchung der MAT-Anweisungen gegenüber einer logisch gleichen Befehlsfolge mit Elementaranweisungen.

```
PROGRAM DUMP

STATEMENT NUMBER      10  INTERNAL LOCATION =    3700
          -19

STATEMENT NUMBER      15  INTERNAL LOCATION =    3699
          -12

STATEMENT NUMBER      20  INTERNAL LOCATION =    3698
           -7        -9          1

STATEMENT NUMBER      30  INTERNAL LOCATION =    3695
           -1         2          1         -7        -14          1         -1          2         -20  196861300
          -14         2         -6          5         -3         55

STATEMENT NUMBER      35  INTERNAL LOCATION =    3679
           -1         4         98          1         -7         -1         -8         98

STATEMENT NUMBER      40  INTERNAL LOCATION =    3671
           -1         6        124          1         -7        316         -5         -1         -8        124

STATEMENT NUMBER      45  INTERNAL LOCATION =    3661
           -1         4         14         -7        315         -1         -9         14

STATEMENT NUMBER      50  INTERNAL LOCATION =    3653
           -3        20

STATEMENT NUMBER      55  INTERNAL LOCATION =    3651
          -16         3   69112940  197644057  200590357        -67        -16          3   23379084  102750251
    200590357       -16          3   90293612  199206152  200590357        -18

STATEMENT NUMBER      60  INTERNAL LOCATION =    3634
           -1         2         14         -7        -17        -67         -1          4         98         14
           -7        -4        -17         -1         16         14         -7        124         -3         99
          315        -5         -2         14         -7         14         -7        315         -2         -3
           -4       -17        -18

STATEMENT NUMBER      65  INTERNAL LOCATION =    3601
          -11

STATEMENT NUMBER      70  INTERNAL LOCATION =    3600

STATEMENT NUMBER      75  INTERNAL LOCATION =    3600

STATEMENT NUMBER      80  INTERNAL LOCATION =    3600
          -11
```

```
PROGRAM DUMP

STATEMENT NUMBER      10  INTERNAL LOCATION =    3700
          -19

STATEMENT NUMBER      20  INTERNAL LOCATION =    3699
          -12

STATEMENT NUMBER      30  INTERNAL LOCATION =    3698
          -28
             1

STATEMENT NUMBER      40  INTERNAL LOCATION =    3696
          -28
             2

STATEMENT NUMBER      50  INTERNAL LOCATION =    3694
           -1          1          315          -8          324          -1           1          327          -8          325
           -1          3          324         326           -2          -9           9          -15           9          324
         3664

STATEMENT NUMBER      55  INTERNAL LOCATION =    3673
           -7         -1            4           1            9          -7          -19

STATEMENT NUMBER      60  INTERNAL LOCATION =    3666
           -4       3677          -12

STATEMENT NUMBER      65  INTERNAL LOCATION =    3663
           -1          1          315          -8          328          -1           1          331          -8          329
           -1          3          328         330           -2          -9           9          -15           9          328
         3604

STATEMENT NUMBER      70  INTERNAL LOCATION =    3642
           -1          1          315          -8          332          -1           1          335          -8          333
           -1          3          332         334           -2          -9          10          -15          10          332
         3607

STATEMENT NUMBER      75  INTERNAL LOCATION =    3621
           -7         -1            9           2            9          -7          29           -3          10           -7
           -1        -19

STATEMENT NUMBER      80  INTERNAL LOCATION =    3609
           -4       3625          -12

STATEMENT NUMBER      90  INTERNAL LOCATION =    3606
           -4       3646          -12

STATEMENT NUMBER      95  INTERNAL LOCATION =    3603
          -11
```



```
CONSTANTS

LOCATION        VALUE
     314            0.00000
     315            1.00000
     316            2.00000
     317            3.00000
     318            4.00000
     319            5.00000
     320            6.00000
     321            7.00000
     322            8.00000
     323            9.00000
     324            0.00000
     325            0.00000
     326            1.00000
     327           25.00000
     328            0.00000
```

Während die Statements 30 und 40 die Speicherplatzadressen 3698 bis 3695 beanspruchen, werden bei Verwendung von Elementaranweisungen für das Einlesen eines Vektors die Speicherplatzadressen 3694 bis 3664 und für das Einlesen einer Matrix die Adressen 3663 bis 3602 belegt.

34 Gesamtüberblick der BASIC-Sprachanweisungen

Die folgende Gleichung fasst die in Band 2 und 3 behandelten Sonderanweisungen in alphabetischer Ordnung zusammen. Diejenigen Anweisungen, die noch nicht in das vorhandene BASIC-Gleichungssystem integriert sind, enthalten anstatt der Definitionsgleichungsnummer die Abschnittsnummer, in der sie erläutert werden.

⟨Sonderanweisung⟩ : =
385 204

⟨ASK BACKGROUND COLOR-Statement⟩ |
338

⟨ASK BEAM-Statement⟩ |
359

⟨ASK BOUNDS-Statement⟩ |
327

⟨ASK CHARACTER HEIGHT-Statement⟩ |
361

⟨ASK CLIP-Statement⟩ |
339

⟨ASK COLOR-Statement⟩ |
338

⟨ASK COLOR COUNT-Statement⟩ |
338

⟨ASK DEVICE BOUNDS-Statement⟩ |
Abs. 334.121

⟨ASK FL-Statement⟩ |
290

⟨ASK JUSTIFY-Statement⟩ |
360

⟨ASK LINE STYLE-Statement⟩ |
338

⟨ASK POSITION-Statement⟩ |
358

⟨ASK STYLE COUNT-Statement⟩ |
338

⟨ASK TEXT ANGLE-Statement⟩ |
362

⟨ASK VIEWPORT-Statement⟩ |
328

⟨ASK WINDOW-Statement⟩ |
329

⟨BREAK-Statement⟩ |
382

⟨CALL-Statement⟩ |
297

⟨CALL PLOT-Statement⟩ |
367

⟨CASE-Statement⟩ |
Abs. 238.2

⟨CHAIN-Statement⟩ |
316

⟨CLEAR-Statement⟩ |
336

⟨CLOSE FL-Statement⟩ |
271

⟨COLLATE-Statement⟩ |
Abs. 325.3

⟨DEBUG OFF-Statement⟩ |
378

⟨DEBUG ON-Statement⟩ |
377

⟨DECLARE-Statement⟩ |
Abs. 325.3

⟨DEF-Block⟩ |
Abs. 325.2

⟨DEF-Statement⟩ |
Abs. 325.2

⟨DEF END-Statement⟩ |
Abs. 325.2

⟨DEF EXIT-Statement⟩ |
Abs. 325.2

⟨DELETE FL-Statement⟩ |
Abs. 332.45

⟨DO EXIT-Statement⟩ |
Abs. 328.1

⟨DO/LOOP-Block⟩ |
Abs. 328.1

⟨DUMP-Statement⟩ |
384

⟨ERASE FL-Statement⟩ |
272

⟨FOR EXIT-Statement⟩ |
Abs. 328.1

⟨FOR/NEXT-Block⟩ |
Abs. 328.1

⟨FUNCTION-Block⟩ |
Abs. 333.13

⟨FUNEND-Statement ⟩ |
Abs. 333.13

⟨FUNEXIT-Statement⟩ |
Abs. 333.13

⟨GRAPHIC INPUT-Statement⟩ |
364

⟨GRAPHIC PRINT-Statement⟩ |
356

⟨IF EOF FL-Statement⟩ |
294

⟨INPUT FL-Statement⟩ |
284

⟨INQUIRE BOUNDS-Statement⟩ |
327

⟨INQUIRE BRUSH-Statement⟩ |
338

⟨INQUIRE CLIP-Statement⟩ |
339

⟨INQUIRE FL-Statement⟩ |
290

⟨INQUIRE POSITION-Statement⟩ |
350

⟨INQUIRE VIEWPORT-Statement⟩ |
328

⟨INQUIRE WINDOW-Statement⟩ |
329

⟨LINE INPUT FL-Statement⟩ |
284

⟨LINK-Statement⟩ |
317

⟨LOOP-Statement⟩ |
Abs. 325.1

⟨MARGIN-Statement⟩ |
Abs. 327

⟨MARGIN FL-Statement⟩ |
Abs. 332.35

⟨MAT FILL-Statement⟩ |
350

⟨MAT INPUT-Statement⟩ |
227

⟨MAT INPUT FL-Statement⟩ |
286

⟨MAT let-Statement⟩ |
237

⟨MAT LET-Statement⟩ |
229

⟨MAT LINE INPUT-Statement⟩ |
227c

⟨MAT LINE INPUT FL-Statement⟩ |
286

⟨MAT OUTPUT FL-Statement⟩ |
281

⟨MAT OUTPUT USING FL-Statement⟩ |
282

⟨MAT PLOT-Statement⟩ |
346

⟨MAT PRINT-Statement⟩ |
249

⟨MAT PRINT FL-Statement⟩ |
281

⟨MAT PRINT USING-Statement⟩ |
257

⟨MAT PRINT USING FL-Statement⟩ |
282

⟨MAT READ-Statement⟩ |
210

⟨MAT READ FL-Statement⟩ |
286

⟨MAT REWRITE FL-Statement⟩ |
Abs. 332.43

⟨MAT REWRITE WITH FL-Statement⟩ |
Abs. 332.44

⟨MAT READ WITH FL-Statement⟩ |
Abs. 332.54

⟨MAT WRITE FL-Statement⟩ |
281

⟨MAT WRITE WITH FL-Statement⟩ |
282

⟨OPEN FL-Statement⟩ |
260

⟨OPTION-Statement⟩ |
Abs. 325.3

⟨OUTPUT FL-Statement⟩ |
274

⟨OUTPUT USING FL-Statement⟩ |
280

⟨PICTURE-Block⟩ |
372

⟨PICTURE-Statement⟩ |
373

⟨PICTURE END-Statement⟩ |
374

⟨PICTURE EXIT-Statement⟩ |
373b

⟨PLOT-Statement⟩ |
342

⟨PRINT ALL-Statement⟩ |
383

⟨PRINT FL-Statement⟩ |
280

⟨PRINT USING FL-Statement⟩ |
280

⟨READ FL-Statement⟩ |
284

⟨READ WITH FL-Statement⟩ |
Abs. 332.52

⟨REDIM-Statement⟩ |
Abs. 331

⟨RESET FL-Statement⟩ |
289

⟨RESTORE FL-Statement⟩ |
289

⟨REWRITE FL-Statement⟩ |
Abs. 332.41

⟨REWRITE WITH FL-Statement⟩ |
Abs. 332.42

⟨SCRATCH FL-Statement⟩ |
272

⟨SELECT-Block⟩ |
Abs. 328.2

⟨SELECT-Statement⟩ |
Abs. 328.2

⟨SELECT END-Statement⟩ |
Abs. 328.2

⟨SET BACKGROUND COLOR-Statement⟩ |
334

⟨SET BEAM-Statement⟩ |
352

⟨SET BOUNDS-Statement⟩ |
321

⟨SET BRUSH-Statement⟩ |
334

⟨SET CHARACTER HEIGHT-Statement⟩ |
354

⟨SET CLIP-Statement⟩ |
335

⟨SET COLOR-Statement ⟩ |
334

⟨SET JUSTIFY-Statement⟩ |
353

⟨SET LINE STYLE-Statement⟩ |
334

⟨SET POSITION-Statement⟩ |
351

⟨SET TEXT ANGLE-Statement⟩ |
355

⟨SET VIEWPORT-Statement⟩ |
324

⟨SET WINDOW-Statement⟩ |
325

⟨SUB-Block⟩ |
306

⟨SUB-Statement⟩ |
308

⟨SUBEND-Statement⟩ |
314

⟨SUBEXIT-Statement⟩ |
313

⟨TEMPLATE-Statement⟩ |
Abs. 332.11

⟨TRACE OFF-Statement⟩ |
381

⟨TRACE ON-Statement⟩ |
380

⟨USE-Statement⟩ |
Abs. 328.3

⟨WHEN/USE-Block⟩ |
Abs. 328.3

⟨WHEN-Statement⟩ |
Abs. 328.3

⟨WHEN END-Statement⟩ |
Abs. 328.3

⟨WRITE FL-Statement⟩ |
280

⟨WRITE WITH FL-Statement⟩ |
280

⟨ZONEWIDTH-Statement⟩ |
Abs. 327

⟨ZONEWIDTH FL-Statement⟩
Abs. 332.36

Zusammen mit den in Gleichung 206 auf Seite 47 in Band 2 aufgeführten Elementaranweisungen bilden die Sonderanweisungen den zur Zeit genormten Gesamtumfang der BASIC-Sprachanweisungen. Die in Band 1 und 3 behandelten BASIC-Sprachanweisungen gehören generell zu den aus Abbildung 22 ersichtlichen Kategorien. Aufgrund dieser Darstellung wird auch deutlich, dass der – an sich willkürlichen – Aufteilung in Elementar- und Sonderanweisungen nur subsidär Bedeutung zukommt.

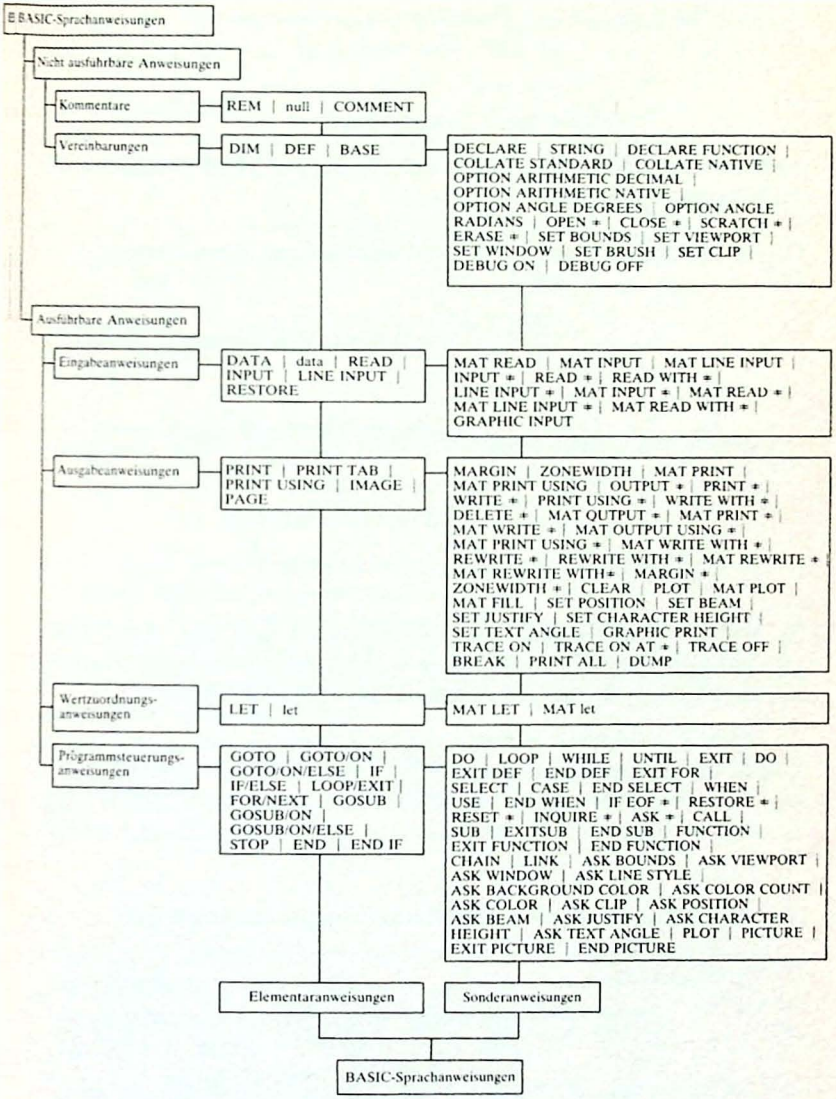


Abbildung 22. Gesamtüberblick zu den BASIC-Sprachanweisungen

Aufgrund von Abbildung 1 ergibt sich für die BASIC-Systemanweisungen folgende Einteilung:

⟨BASIC-Systemanweisung⟩ 401 : = ⟨Systemsteuerungsanweisung⟩ 406 |

⟨Programmanipulationsanweisung⟩ 411 |

⟨Bibliothekverwaltungsanweisung⟩ 432 |

⟨Diagnoseanweisung⟩ 375

Die BASIC-Systemanweisungen unterscheiden sich von den BASIC-Sprachanweisungen strukturell durch die fehlende Statementnummer; im übrigen bestehen sie aus Anweisungskennwort und -Inhalt.

⟨BASIC-Systemanweisung⟩ 402 : = ⟨Systemanweisungskennwort⟩ 403 ⟨Systemanweisungsinhalt⟩ 404

Für die FOSBIC-bezogenen Systemanweisungskennworte gilt:

⟨Systemanweisungskennwort⟩ 403 : = FOSBIC 402 | CATALOG | CATLIST | CHANGE | CLEAR | COMPILE | DELETE | EDIT | LINK | LIST | LIST PRINT | LIST PUNCH | LOAD | NOLIST | REFERENCES | RENUMBER | REPLACE | RUN | RUN PRINT | SAVE | UNSAVE | UNSAVE ALL

Generell bestehen damit die BASIC-Kennworte aus Systemanweisungskennworten und Sprachanweisungskennworten.[1]
Bezüglich des Systemanweisungsinhalts ist zu vermerken, dass er keine Variablen, sondern nur Konstante enthalten darf.[2]

⟨Systemanweisungsinhalt⟩ 404 : = ⟨Leerzeichen⟩ 402 | ⟨Konstante⟩ 7 | ⟨Konstante⟩ 17 ⟨Folgekennwort⟩ 405 ⟨Systemanweisungsinhalt⟩ 404

⟨Folgekennwort⟩ 405 : = AT | FROM | ON | TO 404

Zum Beispiel wird durch die alternativ verwendbaren Anweisungen

```
LIST 1250 TO 2000
LIST FROM 1250 TO 2000
```

ein Ausschnitt einer Programmliste von Statementnummer 1250 bis einschliesslich Statementnummer 2000 erzeugt.
Gemeinsam ist allen BASIC-Systemanweisungen (mit einer Ausnahme bezüglich des Einsatzes von LIST und NOLIST[3]), dass sie nicht innerhalb eines BASIC-Programms auftreten dürfen. Sie können ihm nur vorangestellt werden oder nachfolgen.
BASIC-Systemanweisungen der vorerwähnten Art sind im Rahmen des BASIC-Dialogbetriebs generell erforderlich; beim BASIC-Monologbetrieb (Batch-Remote-Betrieb) reichen die Betriebssystemanweisungen des benutzten Computersystems normalerweise vollständig aus, um alle

1 Vgl. Bd. 1, S. 124. Anstelle der dort angeführten Gleichung 73 kann die äquivalente Gleichung

⟨BASIC-Kennwort⟩ 73a : = ⟨Sprachanweisungskennwort⟩ 72 | ⟨Systemanweisungskennwort⟩ 205 403

treten; es erscheint auch sinnvoll, den Ausdruck Systemanweisungskennwort abgekürzt, wie in Gleichung 73, zu verwenden.
2 Zum Begriff der Variablen vgl. Bd. 1, Abs. 323, zur Definition der Konstanten Bd. 1, Abs. 322.
3 Vgl. Abs. 421.

notwendigen Operationen mit BASIC-Programmen durchführen zu können.[4]

41 Systemsteuerungsanweisungen

BASIC-Systemsteuerungsanweisungen bestehen normalerweise aus vier Subsystemen, die der BASIC-Intitialisierung, Programmeditierung, Programmumsetzung und Programmausführung dienen.
Im Monologbetrieb ist die Ablauffolge für die Subsysteme fest vorgegeben, so dass der Anwender keine Möglichkeit zu deren Beeinflussung hat. Vgl. Abbildung 23.

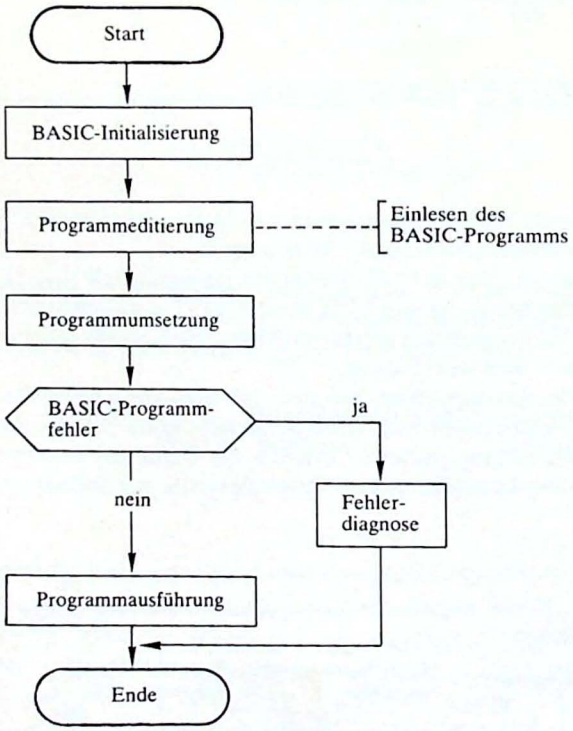


Abbildung 23. BASIC-Systemsteuerung bei Monologbetrieb

4 Zu den Betriebsarten vgl. Bd. 1, Abs. 232, 233.

Im Dialogbetrieb kann jedes Subsystem vom BASIC-Benutzer gezielt angesteuert werden. Sobald das aufgerufene Subsystem seine Bearbeitung beendet hat, wird – im Gegensatz zum Monologbetrieb – nicht automatisch auf das prozedural nachfolgende Subsystem übergegangen, sondern die Steuerung des weiteren Systemablaufs dem BASIC-Benutzer mittels Systemsteuerungsanweisungen überlassen; vgl. Abbildung 24.

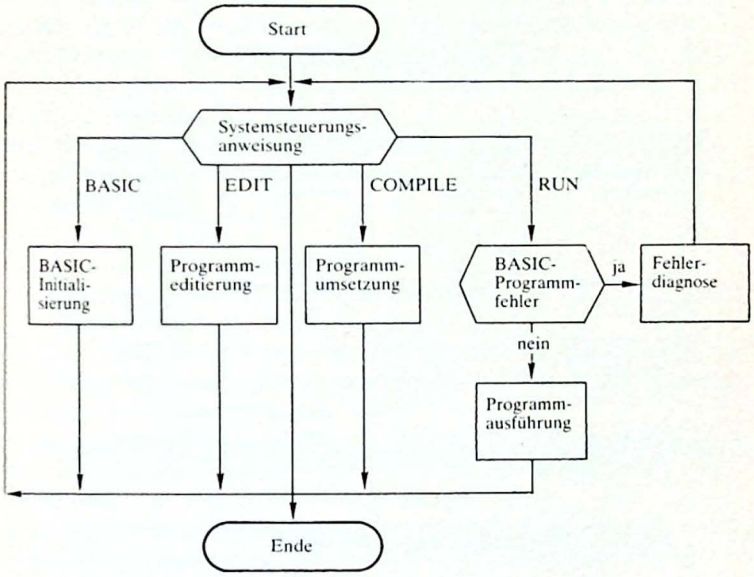


Abbildung 24. BASIC-Systemsteuerung bei Dialogbetrieb

Im folgenden werden die Systemsteuerungsanweisungen nach der aus Abbildung 25 ersichtlichen Gliederung behandelt.

⟨Systemsteuerungs-anweisung⟩ 406 401 : = ⟨BASIC-Initialisierungsanweisung⟩ 407 |

⟨Programmeditierungsanweisung⟩ 408 |

⟨Programmumsetzungsanweisung⟩ 409 |

⟨Programmausführungsanweisung⟩ 410

Die in Abbildung 24 für die Systemsteuerungsanweisungen angegebenen Kennworte BASIC, EDIT, COMPILE und RUN wurden bisher noch nicht standardisiert, finden sich aber in ähnlicher Form bei den meisten dialogorientierten BASIC-Programmiersystemen.
Abbildung 25 zeigt die im Rahmen der FOSBIC Version 2 verfügbaren Systemsteuerungsanweisungen.

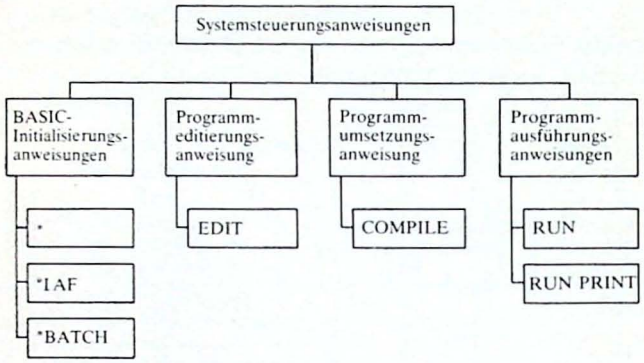


Abbildung 25. Systemsteuerungsanweisungen, FOSBIC Version 2.

411 BASIC-Initialisierungsanweisungen

Verfügt ein Computersystem nur über einen einzigen Sprachprozessor – hier über einen BASIC-Compiler oder BASIC-Interpreter[5] –, so wird er normalerweise automatisch initialisiert; man spricht von impliziter BASIC-Systeminitialisierung.
Sind bei einem Computersystem mehrere Sprachprozessoren vorhanden, so müssen sie meist unter Angabe eines Kennworts aktiviert werden. Die explizite BASIC-Initialisierung hat diesfalls unter Verwendung einer spezifischen Sprachinitialisierungsanweisung zu erfolgen.

5 Zum Begriff des Compilers bzw. Interpreters vgl. Karl Weber und Carl Wolfram Türschmann. FOSBIC-Compiler. Ein BASIC-Compilersystem auf FORTRAN-Basis. Bern 1977, S. 33–34.

⟨BASIC-Initialisierungs-anweisung⟩ : = BASIC | FOSBIC | §§§
407 406

Dabei steht §§§ für weitere systemspezifische Initialisierungskennworte. Bei der FOSBIC Version 2 erfolgt die System- und Sprachinitialisierung grundsätzlich durch das Kennwort FOSBIC; innerhalb des Systems kann durch *IAF oder *BATCH gezielt auf das Dialogsystem (IAF = Interactive Facility) oder Monologsystem (BATCH = Batch-Remote System) zugegriffen werden.[6]
Durch die BASIC-Initialisierung wird der verfügbare Zentralspeicherbereich gelöscht und beim FOSBIC-Compiler automatisch auf Null bzw. Leerzeichen gesetzt.

412 Programmeditierungsanweisungen

BASIC-Programmiersysteme für Dialogbetrieb verfügen zumeist über einen Programmeditor, der ausschliesslich zur Verarbeitung von Programmanipulationsanweisungen[7] dient.
Hochentwickelte Programmeditoren sind in ihrer Leistungsfähigkeit mit guten Texteditoren vergleichbar. Sie dienen zur erleichterten Eingabe und Veränderung von BASIC-Programmen.
Bei Minicomputersystemen wird der verfügbare Programmeditor häufig implizit, in Verbindung mit der BASIC-Systeminitialisierung, aktiviert. Zum expliziten Aufruf des BASIC-Programmeditors dient bei den meisten BASIC-Programmiersystemen das Kennwort EDIT; es sind aber auch andere systemspezifische – in der folgenden Gleichung durch §§§ angedeutete – Kennworte denkbar.

⟨Programmeditierungsanweisung⟩ : = EDIT | §§§
408 406

413 Programmumsetzungsanweisungen

Die Entwicklung von BASIC-Programmen kann im Dialogverkehr unter Verwendung eines BASIC-Editors oder BASIC-Prozessors erfolgen.

6 Vgl. Bd. 1, S. 40–41.
7 Vgl. Abs. 43.

Zur Programmumsetzung in die Maschinensprache ist weiterhin ein BASIC-Interpreter oder BASIC-Compiler erforderlich.
Falls die Programmentwicklung und -Umsetzung mit Hilfe eines BASIC-Interpreters erfolgt, wird jede neu eingegebene BASIC-Programmzeile eigenständig auf syntaktische Fehler überprüft. Ein BASIC-Interpreter vermag deshalb die sich aus der Verbindung mehrer BASIC-Programmzeilen (Statements) ergebenden syntaktischen Fehler nicht zu erkennen; sie werden erst bei versuchter Programmausführung moniert.
Beispielsweise erkennt der BASIC-Interpreter von Nixdorf[8] eine fehlerhafte Verschachtelung von FOR/NEXT-Schleifen[9] nicht. Auch die linke Seite eines LET-Statements wird nicht kontrolliert, so dass ein fehlerhafterweise dort stehender arithmetischer Ausdruck oder gar eine numerische Konstante nicht erkannt wird.
Unzulänglichkeiten der vorerwähnten Art entfallen bei der Programmentwicklung und -Umsetzung mit Hilfe eines BASIC-Compilers. Von ihm wird jede eingegebene BASIC-Programmzeile sofort compiliert; alle erkennbaren syntaktischen BASIC-Fehler werden – durch mehr oder weniger ausführlich gehaltene – Fehlermeldungen ausgewiesen[10] und können vom BASIC-Anwender sofort korrigiert werden. Weiterhin bieten einzelne BASIC-Compiler auch die Möglichkeit zum Ausdruck sogenannter Cross-Reference-Listen.[11]
Der zu verwendende BASIC-Prozessor ist nur im Dialogverkehr explizit – mit einem einfachen Systemanweisungskennwort – aufzurufen. Vgl. hierzu die nachfolgende Gleichung, bei der die Zeichenfolge §§§ systemspezifische Kennworte bezeichnet.

⟨Programmumsetzungsanweisung⟩ : = COMPILE | §§§
409 406

Derartige Anweisungen sind – hier speziell beim FOSBIC-Compiler – nur dann anzugeben, wenn zuvor mit dem Texteditor gearbeitet wurde oder wenn bei der BASIC-Systeminitialisierung implizit kein Interpreter oder Compiler angesprochen wird.

8 Nixdorf. System-Software Business-BASIC. Paderborn 1979, S. 10.59–61.
9 Vgl. Bd. 1, S. 230.
10 Für die FOSBIC Version 1 vgl. Weber und Türschmann, FOSBIC-Compiler, S. 83–91.
11 Vgl. Abs. 431.

414 Programmausführungsanweisungen

Im Monologbetrieb (Batch-Remote System) wird dem BASIC-Prozessor durch das END-Statement[12] angezeigt, dass die Programmumwandlung beendet ist und die Programmausführung gestartet werden kann, falls das übersetzte BASIC-Programm fehlerfrei ist und keine BASIC-Unterprogramme folgen.[13]
Im Dialogbetrieb muss dem BASIC-Programmiersystem die Anweisung zur Programmausführung eigens mitgeteilt werden.[14] Damit wird zunächst eine Sortierung des BASIC-Programms in aufsteigender Reihenfolge der Statementnummern eingeleitet.[15] Darauf erfolgt eine erneute Programmumsetzung, die automatisch zur Programmausführung übergeht, wenn das sortierte BASIC-Programm einschliesslich eventueller BASIC-Unterprogramme fehlerfrei ist. Bei den meisten BASIC-Programmiersystemen wird die Programmausführung durch die RUN-Anweisung gestartet.

⟨Programmausführungsanweisung⟩ 410 : = RUN | §§§ 406

Die Zeichenfolge §§§ steht wiederum für systemspezifische Kennworte.

42 Programmanipulationsanweisungen

Programmanipulationsanweisungen sind generell nur für den BASIC-Dialogbetrieb verfügbar. Sie erlauben es dem BASIC-Benutzer, die im Zentralspeicher residenten BASIC-Programme aufzulisten, zu korrigieren und zu löschen. Damit ergibt sich für die Programmanipulationsanweisungen folgende Strukturgleichung:

⟨Programmmanipulationsanweisung⟩ 411 401 : = ⟨Ausgabeanweisung⟩ 412 | 416 | ⟨Korrekturanweisung⟩ 420 | 426 | ⟨Löschanweisung⟩ 428 | 430

12 Vgl. Bd. 1, Abs. 332.46.
13 Vgl. Bd. 2, Abs. 333.1, S. 168–184.
14 Vgl. Abbildung 24, S. 161.
15 Vgl. Bd. 1, S. 126.

Ansätze zur Standardisierung dieser Anweisungen finden sich in ANSI BASIC.[16] Sie werden dort als »editing commands« bezeichnet.[17]
Die Programmanipulationsanweisungen sind meist in den Programmeditor integriert.[18] Sie führen im allgemeinen zu keiner syntaktischen Überprüfung der bearbeiteten BASIC-Statements. Ausnahmsweise finden geringfügige Kontrollen statt, indem etwa die eingegebene Statementnummer auf Einhaltung der systemspezifischen Grenzen überprüft wird.
Benutzerfreundliche BASIC-Programmiersysteme bieten zur Programmanipulation die Wahlmöglichkeit zwischen Editor- und Compilereinsatz. Programmeditoren sind im allgemeinen mit Programmanipulationsanweisungen besonders gut dotiert und arbeiten sehr schnell; eine genaue syntaktische Analyse der einzelnen BASIC-Statements wird durch die langsamer arbeitenden BASIC-Compiler sichergestellt, deren Programmanipulationsanweisungen im Vergleich zu denjenigen der Programmeditoren üblicherweise allerdings weniger versatil sind.
Bei der FOSBIC Version 2 sind die aus Abbildung 26 ersichtlichen Programmanipulationsanweisungen bei Editor- und Compilereinsatz verfügbar; sie unterscheiden sich nur durch ihre unterschiedliche Verarbeitungsgeschwindigkeit und den syntaktischen Prüfungsumfang.

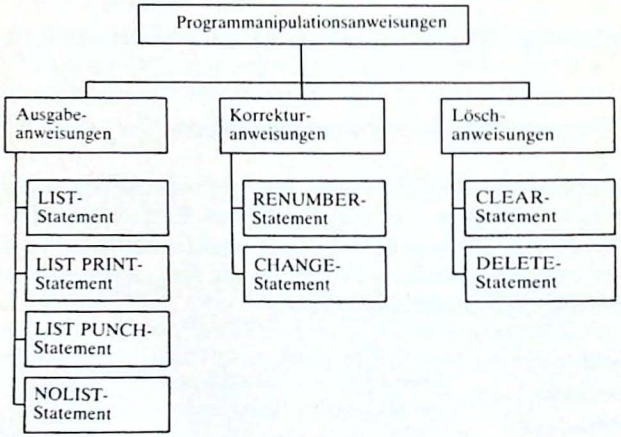


Abbildung 26. Programmanipulationsanweisungen, FOSBIC Version 2.

16 ANSI BASIC 79, p. 133–134. – ANSI BASIC 82, p. 201–203.
17 ANSI BASIC 79, p. 133: "Editing commands allow the programmer to manipulate programs by listing, deleting, or renumbering selected portions of the program." – ANSI BASIC 82, p. 202.
18 Vgl. auch Abs. 412.

421 Ausgabeanweisungen

Die Eingabe eines BASIC-Programms muss normalerweise nicht nach Statementnummern geordnet erfolgen;[19] im Dialogverkehr können insbesondere nachträglich einzufügende oder korrigierte BASIC-Programmzeilen jederzeit an das Ende des erstellten BASIC-Programms geschrieben werden.
Die Reorganisation des Programms in aufsteigender Statementnummernfolge geschieht im Dialogverkehr durch eine LIST-Anweisung. Es handelt sich hierbei um eine terminalbezogene Ausgabeanweisung; sie bewirkt die Programmreorganisation und -Auflistung nach aufsteigenden Statementnummern.[20]
Nach ANSI BASIC ist als Ausgabeanweisung lediglich die Form des LIST-Statements nach Gleichung 412 vorgesehen.[21]

⟨Ausgabeanweisung⟩ : = ⟨LIST-Statement⟩
412 411 413

Dieses List-Statement weist nachfolgende Standardform

⟨LIST-Statement⟩ : =
413 412

LIST |
LIST ⟨Statementbereich⟩ 414

mit

⟨Statementbereich⟩ : = ⟨Bereichsbegrenzer⟩ \|
414 413 415

⟨Bereichs-begrenzer⟩ TO ⟨Bereichs-begrenzer⟩
415 415

auf.

19 Vgl. Bd. 1, Abs. 326.
20 ANSI BASIC 79, p. 132: "Editing facilities enable the lines of a program to be entered in an arbitrary order. ... The program-lines in an unsorted-program shall be sorted into ascending line-number sequence. In the case of duplicate line-numbers, only the textually last program-line with a given line-number shall be retained." – ANSI BASIC 82, p. 201.
21 ANSI BASIC 79, p. 133. – ANSI BASIC 82, p. 202.

Als Bereichsbegrenzer ist neben der Statementnummer auch der Angabe FIRST und LAST zulässig;[22] die Bereichsbegrenzer dürfen sich logisch nicht widersprechen.

⟨Bereichs-begrenzer⟩ : = ⟨Statement-nummer⟩ | FIRST | LAST
415 414 71

Entfällt in einem LIST-Statement die Angabe des Statementbereichs, so erfolgt die Auflistung des gesamten BASIC-Programms.[23] Bei Angabe einer unteren und oberen Bereichsgrenze wird der dazwischen liegende Teil des BASIC-Programms, einschliesslich der Bereichsgrenzen, aufgelistet. Besteht die Statementbereichsangabe nur aus einer Statementnummer, oder der Angabe FIRST oder LAST, so wird nur die entsprechende BASIC-Programmzeile angezeigt.[24]

Die meisten BASIC-Programmiersysteme liegen mit ihren LIST-Anweisungen bereits weitgehend innerhalb der ANSI-Norm. Die FOSBIC Version 2 erfüllt sie vollständig.

Zu beachten ist, dass die Auflistung des gewünschten BASIC-Programms oder Programmteils normalerweise auf demjenigen peripheren Gerät erfolgt, von dem der LIST-Befehl eingegeben wurde.

Abweichungen vom ANSI-Standard können in Einzelfällen insofern bestehen, als die Verwendung von FIRST und LAST als Bereichsbegrenzung nicht zulässig ist. Weiterhin führt die Angabe nur einer Statementnummer als Bereichsbegrenzung bei einzelnen BASIC-Programmiersystemen zur Auflistung des BASIC-Programms ab der angegebenen Statementnummer bis zum Programmende.

Im übrigen können auch bezüglich der Struktur des LIST-Statements bei einzelnen Programmiersystemen Abweichungen vom ANSI-Standard bestehen. Der nach ANSI BASIC vorgesehenen Anweisung

22 ANSI BASIC 79, p. 133. – Der Statementbereich wird nach ANSI BASIC als „segment-specifier", der Bereichsbegrenzer als „segment item" bezeichnet. – ANSI BASIC 82, p. 202.

23 ANSI BASIC 79, p. 133: "The list command shall cause the text of the specified collection of lines to be printed: if the segment-specifier is omitted, the text of the entire program shall be printed." – ANSI BASIC 82, p. 203.

24 Vgl. ANSI BASIC 79, p. 133: "A segment-specifier specifies a collection of lines in a program: if the segment-specifier is a single segment-item, then the collection shall consist of the indicated line; if the segment-specifier contains two segment-items, then the collection shall consist of all lines whose line-numbers are at least as large as the first, but not more than the second." – ANSI BASIC 82, p. 202.

```
LIST 1000 TO 2000
```

entsprechen die Statements

```
1000 LIST 2000
LIST 1000,2000
LIST 1000;2000
```

bei Nixdorf (Business BASIC 8870/1), Sperry Rand (Sperry UNIVAC 1100 Series BASIC) resp. LOGABAX (LX BDOS BASIC-N).[25]
Bei der FOSBIC Version 2 sind neben dem einfachen LIST-Statement auch noch zusätzliche Anweisungsformen zulässig. Vgl. hierzu auch Abbildung 26.

⟨Ausgabeanweisung⟩ 416 : = ⟨LIST-Statement⟩ 413 | ⟨LIST PRINT-Statement⟩ 417 |

⟨LIST PUNCH Statement⟩ 418 | ⟨NOLIST-Statement⟩ 419

Um bei Dialogbetrieb eine Auflistung der gewünschten Programme auf dem Schnelldrucker zu erhalten, wird von der FOSBIC Version 2 das LIST PRINT-Statement angeboten. Seine Struktur ist bis auf das Kennwort identisch mit dem LIST-Statement.

⟨LIST PRINT-Statement⟩ 417 : = 416

LIST PRINT |

LIST PRINT ⟨Statementbereich⟩ 414

Teilweise verfügen auch andere BASIC-Programmiersysteme über ähnliche Spezialanweisungen zur Erzeugung von Drucklisten. Bei der CD

25 Vgl. Nixdorf, System Software Busines Basic, S. 2.10; Sperry Rand. Sperry UNIVAC 1100 Series. BASIC. Rev. 2. St. Paul 1978, p. 4.3; Société LOGABAX. LX 500 BDOS BASIC-N Manuel de Référence. Evry 1980, p. 1.31.

BASIC Version 3 (unter NOS 1 oder NOS/BE 1) muss zum Beispiel das BASIC-Programm durch das Statement

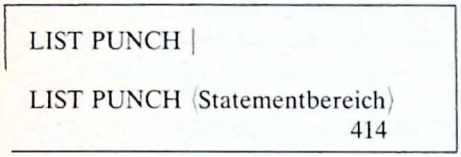
LIST, F=⟨Name⟩

auf ein lokales File geschrieben und von dort durch

ROUTE, ⟨Name⟩, DC=LP,ID=0

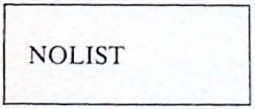
auf ein peripheres Gerät (Schnelldrucker) übertragen werden.
Das bei der FOSBIC Version 2 ebenfalls verfügbare

⟨LIST PUNCH-Statement⟩ : =
417 416



unterscheidet sich vom LIST-Statement nur durch das Kennwort.
Seine Angabe erzeugt die Ausgabe des vorhandenen BASIC-Programms auf 80-Spalten-Lochkarten.
Mit der LIST PUNCH-Anweisung kann der Benutzer seine BASIC-Programme vom Dialogbetrieb ausgehend für den Monologbetrieb auf Lochkartenbasis bereitstellen.
Bei der FOSBIC Version 2 dient das

⟨NOLIST-Statement⟩ : =
419 416



dazu, die Auflistung eines BASIC-Programms im Monologbetrieb zu unterdrücken. Der Anwender schützt dadurch sein BASIC-Programm vor fremder Einblicknahme. Die Auflistung wird jeweils mit einem NOLIST-Statement unterbrochen, kann aber mit einem subsequent in das BASIC-Programm eingeschobenen LIST-Statement wieder aufgenommen werden. Die Verwendung von LIST und NOLIST ohne Statementbereich ist hier ausnahmsweise auch innerhalb eines BASIC-Programms erlaubt.[26]

26 Vgl. S. 159.

422 Korrekturanweisungen

Um ganze BASIC-Programme, Programmteile oder einzelne Programmzeilen einfach verändern bzw. korrigieren zu können, bieten einige BASIC-Programmiersysteme Korrekturanweisungen an, die teilweise einem Texteditor vollständig gleichkommen.
Als Korrekturanweisung ist nach ANSI BASIC nur das RENUMBER-Statement vorgesehen.

⟨Korrekturanweisung⟩ : = ⟨RENUMBER-Statement⟩
420 411 421 | 425

Mit einem RENUMBER-Statement können die Statementnummern eines BASIC-Programms neu festgelegt werden. Es weist nach ANSI BASIC folgende Struktur auf:

⟨RENUMBER-Statement⟩ : =
421 420

```
RENUMBER    |

RENUMBER  ⟨Statementbereich⟩  ⟨Renumberparameter⟩
                    414                422
```

Die Renumberparameter geben an, mit welcher Statementnummer die Neunumerierung beginnen und mit welcher Schrittweite sie fortgesetzt werden soll. Es gilt:

⟨Renumberparameter⟩ : = AT ⟨Anfangsnummer⟩ |
422 421 423

AT ⟨Anfangsnummer⟩ STEP ⟨Schrittweite⟩
mit 423 424

⟨Anfangsnummer⟩ : = ⟨Statementnummer⟩
423 422 71

⟨Schrittweite⟩ : = ⟨Vorzeichenlose Ganzzahl > 0 ⟩
424 422 23

Fehlt die Angabe der Renumberparameter in einem RENUMBER-Statement, so soll als Anfangsnummer implizit 100 und als Schrittweite 10 angenommen werden.[27] Dies gilt für einen angegebenen Statementbereich oder, wenn diese Spezifikation fehlt, für das ganze BASIC-Programm.
Zu beachten ist, dass durch die RENUMBER-Anweisung auch die in Programmverzweigungs- und Programmkoppelungsanweisungen[28] als Sprungziel angegebenen Statementnummern mit umgesetzt werden.
Durch die Anweisung

```
RENUMBER AT 1000 STEP 20
```

wird das gesamte BASIC-Programm neu durchnumeriert, wobei die erste Programmzeile die Statementnummer 1000, die zweite die Statementnummer 1020 usw. erhält.
Mit der Anweisung

```
RENUMBER 200 TO 400 AT 250 STEP 5
```

wird der Programmteil zwischen den Statementnummern 200 bis 400 einschliesslich neu numeriert, wobei mit der Statementnummer 250 begonnen wird und die nachfolgenden Programmzeilen jeweils eine um 5 erhöhte Statementnummer erhalten.
Versucht ein neu numeriertes Statement eine ausserhalb des umzustellenden Bereichs liegende Anweisung aufgrund derselben Statementnummer zu überschreiten, so kommt es zum Ausführungsabbruch; das Ursprungsprogramm bleibt diesfalls aber erhalten.[29]

27 ANSI BASIC 79, p. 133–134: "The renumber command shall cause new line-numbers to be assigned to all lines in the specified collection (or in the entire program if the segment-specifier is omitted). The new line-number of the first line in the collection shall be the initial-number (100 if no initial-number is specified); line-numbers for succedding lines in the collection shall be generated using the specified step-size (or 10 if no step-size is specified). The line-numbers in control statements and formatted-print-lists throughout the entire program shall be changed to conform to the new line-numbers." – ANSI BASIC 82, p. 203.

28 Vgl. Bd. 1, S. 205–244.

29 Fehlerliste nach ANSI BASIC 79, p. 134: "Renumber sequence-parameters are chosen so that renumbered lines would overlay or surround lines not being renumbered (fatal). A renumber would resolve an unresolved line-number reference in a control statement (fatal). A renumber would create a line-number higher than the host implementation could accept (fatal). A renumber would cause a change in the sequence order of lines (fatal)." – ANSI BASIC 82, p. 203.

Bei der FOSBIC Version 2 steht das RENUMBER-Statement in der nach ANSI BASIC vorgesehenen Form zur Verfügung. Andere BASIC-Programmiersysteme weichen vom ANSI BASIC-Standard noch erheblich ab; so wird etwa bei (UNIVAC 1100 Series BASIC) RESEQUENCE anstelle des Kennworts RENUMBER benutzt.[30] Weiterhin lassen die meisten BASIC-Programmiersysteme nur eine Neunumerierung des gesamten BASIC-Programms zu, wobei häufig die Anweisungsstruktur

⟨RENUMBER-Statement⟩ : =
425 420

RENUMBER |
⟨Anfangsnummer⟩ RENUMBER ⟨Schrittweite⟩
423 424

verlangt wird.[31]
Bei der FOSBIC Version 2 gilt für die Korrekturanweisungen

⟨Korrekturanweisung⟩ : = ⟨RENUMBER-Statement⟩ | ⟨CHANGE-Statement⟩
426 411 421 427

Das FOSBIC-spezifische CHANGE-Statement unterliegt der Strukturformel

⟨CHANGE-Statement⟩ : =
427 426

CHANGE / ⟨Stringinhalt⟩ / ⟨Stringinhalt⟩ /
34 34

Das CHANGE-Statement bewirkt im zuletzt eingegebenen oder angezeigten BASIC-Statement eine Stringkorrektur. Dabei wird der im CHANGE-Statement zuerst angeführte Stringinhalt durch den an zweiter Stelle angegebenen Stringinhalt ersetzt.

30 Sperry Rand. Sperry UNIVAC 1100 Series. BASIC. Rev. 2. St. Paul 1978, p. 5.5.
31 Vgl. etwa Nixdorf, System-Software Business-Basic, S. 2.13–14.

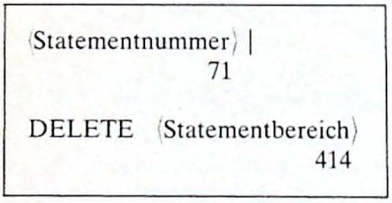
423 Löschanweisungen

Durch Löschanweisungen kann ein gesamtes BASIC-Programm oder ein bestimmter Teil davon gelöscht werden. Der gelöschte Programmteil geht verloren, sofern das BASIC-Programm nicht vor der Löschanweisung durch eine SAVE-Anweisung[32] auf einen externen Speicher gerettet wurde.
Nach ANSI BASIC ist nur eine Löschanweisung vorgesehen.

⟨Löschanweisung⟩ : = ⟨DELETE-Statement⟩
428 411 429

Das DELETE-Statement kann in impliziter oder expliziter Form verwendet werden, wobei sich die letztere nur im Kennwort von dem nach ANSI BASIC vorgesehenen LIST-Statement unterscheidet.[33]

⟨DELETE-Statement⟩ : =
429 428



Durch das implizite DELETE-Statement (ohne Kennwortangabe) kann lediglich ein Statement aus dem BASIC-Programm eliminiert werden; das explizite DELETE-Statement lässt die Löschung eines oder mehrerer Statements, bis hin zum Gesamtprogramm, zu.
Besteht der Statementbereich beim expliziten DELETE-Statement nur aus einer Statementnummer, so wird nur die referierte Programmzeile gelöscht.[34] Umfasst der Statementbereich eine untere und eine obere Begrenzung, so wird der gesammte Programmteil zwischen unterer und oberer Bereichsgrenze, einschliesslich der begrenzenden Statements, aus

32 Vgl. Abs. 431.
33 Vgl. Abs. 421.
34 Zur fehlerhaften Statementangabe vgl. ANSI BASIC 79, p. 134: "Implementations may issue a warning if a non-existent line-number is referenced by the segment-specifier for a deletion." – ANSI BASIC 82, p. 203.

dem BASIC-Programm eliminiert. Da nach ANSI BASIC die Bereichsgrenze auch mit FIRST und LAST angegeben werden kann, wird durch die Anweisung

```
DELETE FIRST TO LAST
```

das gesamte vorhandene BASIC-Programm gelöscht.[35]
Bezüglich des DELETE-Statements finden sich bei einzelnen BASIC-Programmiersystemen insofern Besonderheiten, als das Folgekennwort TO durch ein Komma ersetzt wird. Bei UNIVAC 1100 Series BASIC können mit einem DELETE-Statement mehrere Programmteile gelöscht werden.[36] So bewirkt die Anweisung

```
EDIT DELETE 10,50, 100,200, 2000
```

dass die Statements von 10 bis 50, 100 bis 200 und 2000 bis zum letzten Programmstatement gelöscht werden. Interessant ist bei diesem System auch das sogenannte EXTRACT-Statement, das eine komplementäre Funktion zum DELETE-Statement beinhaltet, indem es alle BASIC-Programmzeilen löscht, die *nicht* zwischen den angegebenen Bereichsgrenzen liegen.[37] Nach Ausführung der Anweisung

```
EDIT EXTRACT 20, 200, 700
```

sind nur noch die Statementzeilen von 20 bis 200 und ab 700 vorhanden.
Die FOSBIC Version 2 hält sich an den Normvorschlag von ANSI BASIC, lässt aber zusätzlich noch ein CLEAR-Statement zur gesamthaften Löschung eines BASIC-Programms zu. Mithin gilt:

⟨Löschanweisung⟩ : = ⟨DELETE-Statement⟩ | ⟨CLEAR-Statement⟩
430 411 429 431

Das CLEAR-Statement besteht meist nur aus einem Kennwort.

35 ANSI BASIC 79, p. 133: "The delete command shall cause the specified collection of lines to be deleted from a program." – ANSI BASIC 82, p. 203.
36 Sperry Rand. Sperry UNIVAC 1100 Series BASIC, p. 5.4.
37 Sperry Rand. Sperry UNIVAC 1100 Series BASIC, p. 5.5.

⟨CLEAR-Statement⟩ : =
431 430

CLEAR |
NEW

Teilweise wird dieses Statement implizit immer beim Aufruf des BASIC-Programmiersystems oder bei der Angabe einer Ladeanweisung[38] vorab durchgeführt.

43 Bibliothekverwaltungsanweisungen

Die Notwendigkeit, über Dialogterminals eingegebene BASIC-Programme permanent und leicht wiederauffindbar zu speichern, führte zur Entwicklung benutzerorientierter Bibliothekverwaltungsanweisungen. Sie wurden bisher im Rahmen von ANSI BASIC noch nicht standardisiert, finden sich aber in strukturell ähnlicher Form bei allen BASIC-Dialogsystemen.
Im folgenden wird ein in sich geschlossenes, im Rahmen der FOSBIC Version 2 realisiertes, System von Bibliothekverwaltungsanweisungen vorgestellt. Es umfasst die aus Abbildung 27 ersichtlichen vier Statementgruppen.

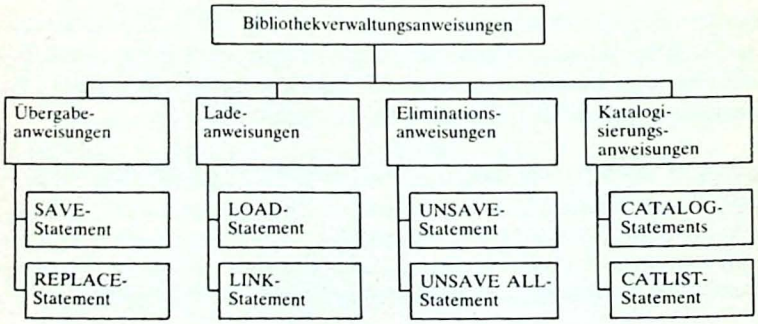


Abbildung 27. Bibliothekverwaltungsanweisungen, FOSBIC Version 2.

38 Vgl. Abs. 432.

Die einzelnen Anweisungen dienen dazu, ein BASIC-Programm oder Teile eines BASIC-Programms
- vom Zentralspeicher auf einen peripheren Speicher zu übernehmen
- von einem peripheren Speicher in den Zentralspeicher zu laden
- die Programmbibliothek teilweise oder vollständig zu eliminieren
- am Bildschirm aufzuzeigen oder eine Programmübersicht der Bibliothek zu erstellen.

Für das in Abbildung 27 dargestellte System von Bibliothekverwaltungsanweisungen gilt:

⟨Bibliothek-verwaltungs-anweisung⟩ 432 401 : = ⟨Übergabe-anweisung⟩ 433 | ⟨Lade-anweisung⟩ 439 | ⟨Eleminations-anweisung⟩ 442 | ⟨Katalogisierungs-anweisung⟩ 445

Vermerkt sei, dass einzelne BASIC-Programmiersysteme über keine besonderen Bibliothekverwaltungsanweisungen verfügen; die Bibliothekverwaltung erfolgt diesfalls unter Verwendung der allgemeinen Systemsoftware.[39]

Weiterhin wird als Peripherspeicher normalerweise die Magnetplatte bevorzugt; Minicomputersysteme gestatten jedoch auch die Programmspeicherung auf Magnetbandkassetten.[40]

Bei der FOSBIC Version 2 können die Bibliothekverwaltungsanweisungen auch im Monologbetrieb eingesetzt werden. So kann mit einem SAVE-Statement ein in Form einer Lochkartendatei vorhandenes BASIC-Programm auf ein Magnetplattenfile geschrieben und von dort im Dialog weiter verwendet werden. Damit gewährleistet die FOSBIC Version 2 eine lückenlose Konvertibilität zwischen Monolog- und Dialogbetrieb. Vgl. Abbildung 28. Sie zeigt, dass der BASIC-Benutzer bei der FOSBIC Version 2 – in Verbindung mit dem LIST PUNCH-Statement[41] – die Möglichkeit zum beliebigen Wechsel zwischen Monolog- und Dialogbetrieb hat.

39 Dies gilt beispielsweise für die CD BASIC-Version gemäss Control Data. BASIC Version 3. Reference. Manual. Sunnyvale 1982, p. 10.1–12.9.

40 Dieses Speichermedium gehört zum Beispiel beim PET Commodore 2001 zur Standardausrüstung und ist in das Dialogterminal eingebaut.

41 Vgl. Abs. 421.

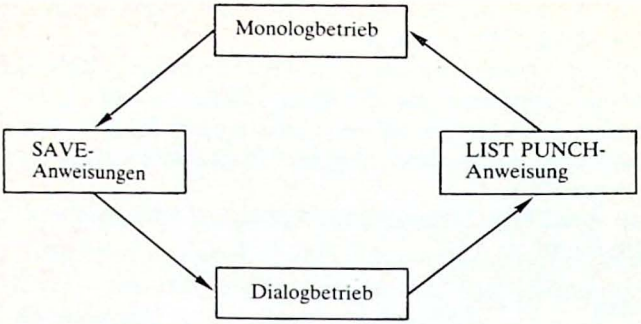


Abbildung 28. Alternativbetrieb, FOSBIC Version 2.

431 Übergabeanweisungen

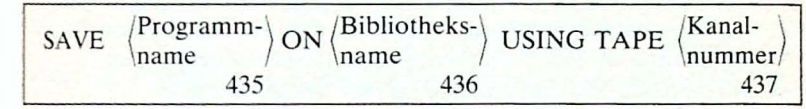
Die Übergabeanweisungen dienen dazu, BASIC-Programme aus dem Zentralspeicher auf einen peripheren Speicher zu übertragen. Die transferierten Programme werden einzeln gespeichert oder gemeinsam in einer Programmbibliothek gesammelt, wobei die Bibliotheksorganisation im einzelnen stark systemabhängig ist.
Die Übergabeanweisungen sind durch ANSI BASIC noch nicht genormt. Unter besonderer Berücksichtigung der FOSBIC Version 2 wird im folgenden zwischen zwei Übergabeanweisungen differenziert, die sich in ähnlicher Form auch bei anderen BASIC-Programmiersystemen finden.

⟨Übergabeanweisung⟩ : = ⟨SAVE-Statement⟩ | ⟨REPLACE-Statement⟩
433 432 434 438

Jede dieser Anweisungen bewirkt, dass auf einem peripheren Speicher eine Kopie des gerade im Zentralspeicher residenten BASIC-Programms erstellt wird. Dabei kann es zur Erstellung eines völlig neuen Programmfiles (SAVE-Statement) oder zur Überschreibung/Ersetzung eines bereits bestehenden Programmfiles (REPLACE-Statement) kommen.
Mit einem SAVE-Statement wird ein Programmfile so gekennzeichnet, dass es sich eindeutig von anderen Programmfiles unterscheidet und jederzeit wieder aufgefunden werden kann. Durch systemabhängige Parameter wird weiterhin der Datenschutz sichergestellt, die Zugriffsberechtigung gesteuert und eventuell auch die Organisationsform des Programmfiles definiert. Weitere Parameter können statistische Werte über

das gespeicherte Programm enthalten (Zugriffszahl, Programmgrösse, letztes Zugriffsdatum, Erstellungsdatum, usw.).
Für die FOSBIC Version 2 wurde die Struktur des SAVE-Statements wie folgt festgelegt:

⟨SAVE-Statement⟩ : =
434 433



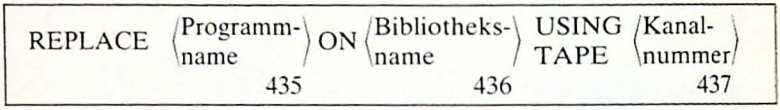
⟨Programmname⟩ : = ⟨Alphanumerische Konstante⟩
435 434 32

⟨Bibliotheksname⟩ : = ⟨Alphanumerische Konstante⟩
436 434 32

⟨Kanalnummer⟩ : = ⟨I-formatierte Ganzzahl⟩
437 434 22

Programm- und Bibliotheksname dürfen maximal 20 Zeichen umfassen. Die Kanalnummer muss eine I-formatierte Ganzzahl zwischen zwei implementationsabhängigen Werten sein. Die Angabe USING TAPE ⟨Kanalnummer⟩ kann weggelassen werden, wenn die standardmässig vorgesehene Kanalnummer verwendet werden soll. In allen anderen Fällen muss die Kanalnummer explizit spezifiziert werden.
Wird ein SAVE-Statement auf ein leeres Plattenfile ausgeführt, so wird damit eine Programmbibliothek mit dem angegebenen Bibliotheksnamen aufgebaut und das im Zentralspeicher residenten BASIC-Programm als erstes Programm unter dem angegebenen Programmnamen in der neu errichteten Bibliothek abgestellt. Durch weitere SAVE-Statements wird das jeweils zu speichernde Programm an das Ende der spezifizierten Bibliothek angefügt. Ist der in einem SAVE-Statement angegebene Programmname in der angesprochenen Bibliothek bereits belegt, so wird dies durch eine Fehlermeldung angezeigt und das SAVE-Statement nicht ausgeführt.
Ein abgespeichertes BASIC-Programm kann nur mit einem entsprechenden REPLACE-Statement überschrieben werden, wobei die Grösse des zu übernehmenden Programms nicht mit der Grösse des ursprünglich gespeicherten Programms übereinstimmen muss.
Für die FOSBIC Version 2 gilt bezüglich des REPLACE-Statements:

⟨REPLACE-Statement⟩ : =
438 433



Existiert der im REPLACE-Statement angegebene Programmname in der spezifizierten Bibliothek nicht, so führt dies auf eine Fehlermeldung; die Ausführung des REPLACE-Statements wird unterbunden.
Für die FOSBIC Version 2 sind folgende Beispiele gültig:

```
SAVE "PROGRAMM-1" ON "TEST-BIBLIOTHEK" USING TAPE=8
REPLACE "PROGRAMM-2" ON "TEST-BIBLIOTHEK"
SAVE "TESTPROGRAMM"
SAVE
SAVE ON "TEST-BIBLIOTHEK"
```

Durch Weglassen des Programmnamens im SAVE-Statement wird ein ungeschütztes – namenloses – Programmfile in der angegebenen Bibliothek abgelegt. Theoretisch kann auch auf Angabe des Bibliotheknamens verzichtet werden. Dies führt jedoch zu einer vollständig ungeschützten Bibliothek, auf die jeder Benutzer zugreifen kann.
Mit SAVE- und REPLACE-Statements können nicht nur vollständige und syntaktisch einwandfreie BASIC-Programme, sondern auch unvollständige und fehlerhafte BASIC-Programmteile in eine Bibliothek übernommen werden.
Bei CD BASIC lauten die entsprechenden Übertragungsanweisungen wie folgt:[42]

SAVE (⟨Programmname⟩)

und

REPLACE (⟨Programmname⟩)

Allerdings unterliegt der Programmname hier den strengeren Bedingungen, dass das erste Zeichen ein Buchstabe sein muss und dass die Gesamtlänge des Programmnamens höchstens 7 Zeichen umfassen darf, wobei alle Sonderzeichen verboten sind.
Bei einigen Minicomputersystemen erfüllt das SAVE-Statement auch die Funktion des REPLACE-Statements, was leicht zu einem versehentlichen Überschreiben eines bestehenden Programmfiles führen kann, wenn ein vorhandener Programmname ein zweites Mal vergeben wird.[43]

42 Vgl. Control Data, BASIC-Version 3. Reference Manual, p. D. 1–5.
43 Vgl. Nixdorf, System-Software Business-Basic, S. 22–26.

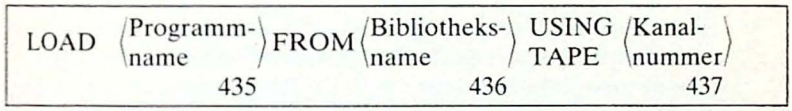
432 Ladeanweisungen

Eine Ladeanweisung bewirkt die Übertragung eines auf einem peripheren Speicher residenten Programms in den Zentralspeicher.
Bei der FOSBIC Version 2 stehen zwei Ladeanweisungen zur Verfügung:

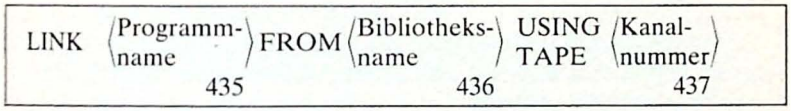
⟨Ladeanweisung⟩ : = ⟨LOAD-Statement⟩ | ⟨LINK-Statement⟩
439 432 440 441

Über die FOSBIC-spezifische Statementstruktur orientieren die folgenden Gleichungen.

⟨LOAD-Statement⟩ : =
440 439



⟨LINK-Statement⟩ : =
441 439



Die LOAD- und LINK-Anweisungen unterscheiden sich strukturell nur bezüglich ihrer Kennworte. In ihrer Wirkungsweise bestehen wesentliche Unterschiede.
Durch eine LOAD-Anweisung wird zunächst der bisher verfügbare Zentralspeicherinhalt gelöscht und dann das aufgerufene BASIC-Programm gelesen. Mehrere BASIC-Programmteile können nicht durch sequentielle LOAD-Anweisungen aufgerufen und miteinander verkettet werden. Hierzu ist die LINK-Anweisung zu verwenden, durch die bereits im Zentralspeicher vorhandene BASIC-Programme nicht gelöscht sondern miteinander verkettet werden. Bei entsprechender Statementnumerierung können auch BASIC-Programmteile in bestehende Programme eingefügt werden.
Die meisten übrigen BASIC-Programmiersysteme verfügen nur über die LOAD-Anweisung und bieten dem Benutzer mithin keine Möglichkeit

zur Verkettung peripher gespeicherter Programme oder Programmteile. Weiterhin unterscheiden sich die Ladeanweisungen anderer BASIC-Programmiersysteme auch bezüglich des verwendeten Kennworts in der LOAD-Anweisung.
Bei UNIVAC 1100 Series BASIC erfolgt das Laden eines gespeicherten BASIC-Programms durch das Statement

OLD: ⟨Programmname⟩

wobei der Programmname maximal 12 Zeichen umfassen darf. Allerdings kann mit diesem System keine Programmbibliothek aufgebaut werden; ebenso ist es nicht möglich, Programmteile zu verketten.[44]
Ähnliches gilt für die LOAD-Anweisung bei CD BASIC.[45]
Mit Business-Basic von Nixdorf erfolgt das Laden eines BASIC-Programms durch

BASIC ⟨Programmname⟩

wobei hier allerdings neben der eigentlichen Programmbezeichnung noch eine eventuell notwendige Angabe der logischen Plattennummer, von der aus das Programm geladen werden soll, im Programmnamen enthalten sein muss.[46]

433 Eliminationsanweisungen

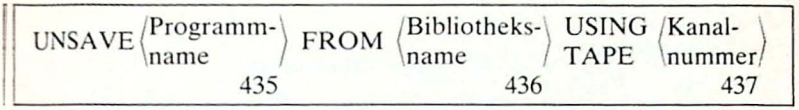
Durch eine Eliminationsanweisung wird ein BASIC-Programm aus einer Programmbibliothek entfernt und eventuell weitergehend auch das gesamte Programmfile gelöscht.
Für die genannten Operationen sind in der FOSBIC Version 2 die folgenden Statements vorgesehen:

⟨Eliminations-anweisung⟩ 442 : = ⟨UNSAVE-Statement⟩ 432 | ⟨UNSAVE ALL-Statement⟩ 443 444

Durch das UNSAVE-Statement wird nur das angegebene BASIC-Programm und dessen Name aus der Programmbibliothek entfernt.

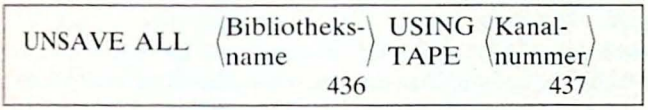
44 Sperry Rand, Sperry UNIVAC 1100 Series. BASIC, p. 5.1.
45 Control Data, BASIC Version 3. Reference Manual, p. 1.11, D. 1–2.
46 Nixdorf, System-Software Business-Basic, S. 12.18–19. – In Verbindung mit dem im Text erwähnten BASIC-Statement kann auch ein LOAD-Statement benutzt werden, falls das zu ladende BASIC-Programm als Textfile gespeichert wurde; vgl. Nixdorf, System-Software Business-Basic, S. 12.18–19, 2.11.

⟨UNSAVE-Statement⟩ : =
443 442



Durch das UNSAVE ALL-Statement wird die gesamte Programmbibliothek, mit Bibliotheksnamen und allen Programmnamen, gelöscht.

⟨UNSAVE ALL-Statement⟩ : =
444 442



Die Anweisungsstruktur der meisten anderen BASIC-Programmiersysteme ist ähnlich. Zentrale Unterschiede bestehen bezüglich der Kennworte und der Art des Verwaltungssystems.
Bei CD BASIC ist als Programmeliminationsanweisung

PURGE ⟨Programmname⟩

zu verwenden;[47] Nixdorf Business-Basic[48] verlangt die Anweisung

KILL ⟨Programmname⟩

Die Zuordnung dieser Anweisungen innerhalb des BASIC-Programmiersystems wird zum Teil sehr unterschiedlich gehandhabt. Bei CD BASIC gehört das PURGE-Statement zur Systemsoftware (Cyber Control Language – CCL). Das KILL-Statement von Nixdorf wird gleichzeitig als System- und als Sprachanweisung zugelassen.

434 Katalogisierungsanweisungen

Die Katalogisierungsanweisungen ermöglichen es dem BASIC-Anwender, sich einen Überblick über den Inhalt eines einzelnen BASIC-Programms, einer BASIC-Programmbibliothek oder sogar einer ganzen Magnetplatte zu verschaffen.

47 Control Data, BASIC Version 3. Reference Manual, p. D. 1.
48 Nixdorf, System-Software Business-Basic, S. 2.22, 2.33.

Die Wirkungsweise der Katalogisierungsanweisungen wird durch den Aufbau der systemspezifischen Programmbibliothek bestimmt.
Für die FOSBIC Version 2 gilt:

⟨Katalogisierungs-anweisung⟩ (445, 432) : = ⟨CATALOG-Statement⟩ (446) |

⟨CATALIST-Statement⟩ (447) |

⟨CATPRINT-Statement⟩ (448)

Das CATALOG-Statement erstellt eine Übersicht der in einer Programmbibliothek gespeicherten BASIC-Programme. Bei seinem Aufruf muss nur der Bibliotheksname und eventuell die Kanalnummer angegeben werden.

⟨CATALOG-Statement⟩ (446) : = (445)

CATALOG ⟨Bibliotheks-name⟩ (436) USING TAPE ⟨Kanal-nummer⟩ (437)

Durch Angabe des Statements

```
CATALOG "TESTFILE-1"
```

wird beispielsweise folgende Übersichtsausgabe der BASIC-Programmbibliothek mit dem Namen »TESTFILE-1« erzeugt:

```
CATALOG OF TAPE  6
FILE NAME=TESTFILE-1
NUMBER OF PROGRAMS=   12
NUMBER OF ACCESSES=  134
LAST DATE OF ACCESS= 81/12/14.
CREATION DATE= 80/01/14.

    PROGRAM NAME          NUMBER OF   NUMBER OF    LAST DATE     CREATION
                          STATEMENTS  ACCESSES     OF ACCESS       DATE

EXPONENT.GLAETTUNG          54          12        81/12/14.     80/01/14.
CRITICAL-PATH-METHOD       110          22        81/12/14.     80/04/14.
PREISELASTIZITAET           41           5        81/12/14.     80/11/12.
OSZILLATIONSMODELL-1       166           7        81/12/14.     80/11/13.
STAT.MASSZAHLEN             48           2        81/12/14.     80/11/17.
LINEARE REGRESSION         110          22        81/12/14.     80/11/22.
INPUT-OUTPUT-ANALYSE       139          17        81/12/14.     80/11/22.
```

```
MULTIPLE REGRESSION        127        13        81/12/14.    80/11/24.
KOSTENABW.ANALYSE          249         3        81/12/14.    80/11/29.
2-PHASEN-METHODE           251        10        81/12/14.    81/01/23.
OSZILLATIONSMODELL-2       169        11        81/12/09.    81/02/10.
MULTISEK.SPIELMODELL        97         5        81/11/11.    81/02/15.

CATALOG FINISHED
```

Das CATLIST-Statement erlaubt es, den Inhalt eines einzelnen BASIC-Programms anzuzeigen, ohne dass dabei ein im Zentralspeicher eventuell vorhandenes BASIC-Programm gelöscht wird. Der BASIC-Anwender kann dadurch während einer Dialogsitzung die in seiner Programmbibliothek enthaltenen BASIC-Programme oder -Programmteile jederzeit einsehen und eventuell Teile davon in sein gerade zu erstellendes Programm (mit Hilfe eines LINK-Statements[49]) übernehmen.
Die Struktur des CATLIST-Statements folgt wieder dem bekannten Aufbau:

⟨CATLIST-Statement⟩ : =
447 445

CATLIST ⟨Programm-name⟩ FROM ⟨Bibliotheks-name⟩ USING TAPE ⟨Kanal-nummer⟩
435 436 437

Das CATPRINT-Statement entspricht in seiner Wirkungsweise dem CATALOG-Statement, wobei die Übersichtsangabe der in der Bibliothek enthaltenen BASIC-Programme nunmehr auf einem Schnelldrucker erfolgt. Seine Struktur unterscheidet sich daher auch nur im Kennwort vom CATALOG-Statement.

⟨CATPRINT-Statement⟩ : =
448 445

CATPRINT ⟨Bibliotheks-name⟩ USING TAPE ⟨Kanal-nummer⟩
436 437

Bei einigen Maschinensystemen steht die CATALOG-Anweisung ausserhalb des BASIC-Programmiersystems und ist in die Systemsoftware integriert. Dies gilt beispielsweise für das CATLIST-Statement der CD

49 Vgl. Abs. 432.

Cyber 174; als CCL-Anweisung (Cyber Control Language) zeigt es nur die abgespeicherten BASIC-Programmfiles, sondern alle unter einer Benutzernummer aufgebauten Programm- und Datenfiles an.[50] Ähnliches gilt auch für das System Nixdorf 8870/1, wo durch ein sogenanntes LIBR-Statement sämtliche Programm- und Datenfiles einer Magnetplatte, die unter einem bestimmten Benutzerkennwort verwendet werden dürfen, angezeigt werden.[51]

44 Diagnoseanweisungen

Zu berücksichtigen sind das DEBUG ON-Statement und das DEBUG OFF-Statement. Zu diesen Systemdiagnoseanweisungen vgl. Abs. 335.

45 Cross-Reference-Listen

BASIC-Programmiersysteme, die über einen BASIC-Compiler oder einen hochentwickelten BASIC-Interpreter verfügen, erlauben meistens die Ausgabe einer sogenannten Cross-Reference-Liste.
Die erzeugten Cross-Reference-Listen sind bezüglich Aufbau und Umfang in besonderem Masse systemabhängig. Die meisten BASIC-Programmiersysteme liefern mit diesen Listen systemspezifische Angaben, wie

- beanspruchter Zentralspeicherplatz
- Anzahl der Statements
- Anzahl der einfachen und indizierten (numerischen und alphanumerischen) Variablen
- implizit und explizit dimensionierte Variable
- systemspezifische Adressen von Variablen (meist in oktaler Form)
- Speicheradressen der Konstanten, usw.

Diese Angaben stellen oft eine maschinenorientierte Wiedergabe zentraler Werte des übersetzten BASIC-Programms dar. Der Zusammenhang zwischen dem ursprünglichen BASIC-Programm und den Daten einer systemspezifischen Cross-Reference-Liste kann dann nur mit speziellen Kenntnissen der Systemsoftware hergestellt werden.
Das System Nixdorf 8870/1 erlaubt es dem Anwender, durch ein spezielles Systemprogramm eine anwendungsorientierte Cross-Reference-Liste

50 Control Data. NOS Version 1. Reference Manual. Vol. 1. Sunnyvale 1979, p. 1.7–4, 1.8–8.
51 Nixdorf. Systemsoftware 8870/1. Bedienerhandbuch. Paderborn 1979, S. 7.24–26.

zu erhalten. Diese Liste unterteilt sich in drei Abschnitte. Der erste Abschnitt gibt eine Übersicht über die im Programm auftretenden Variablen, wobei für jede Variable Name, Typ, Grösse (Dimension) und relevante Statementnummern angegeben werden. Dabei wird jede Verwendung einer Variablen auf der linken Seite einer Gleichung mit einem M und auf der rechten Seite mit einem R vermerkt. Kommt eine Variable in einem Vergleich vor, oder tritt sie als Laufvariable auf, so trägt das Statement keine Kennung.
Ein zweiter Listenabschnitt gibt eine kurze Statistik wieder. In einem dritten Listenabschnitt wird aufgezeigt, welche Statementnummern durch GOTO- und GOSUB-Statements angesprochen werden. Die Liste enthält die Nummer des angesprochenen Statements und die Nummer derjenigen Anweisungen, die auf das betreffende Statement verweisen. Die Nummer wird mit einem T gekenzeichnet, wenn der Verweis von einem GOTO-Statement herrührt; ein S verweist auf ein GOSUB-Statement.
Die so strukturierten Cross-Reference-Listen unterstützen nicht nur den Programmhersteller beim Auffinden von logischen Fehlern, sondern erleichtern es auch fremden Anwendern, sich schnell in die Struktur eines BASIC-Programms hineinzufinden. Cross-Reference-Listen der letzterwähnten Art vermögen deshalb die Transportabilität von BASIC-Programmen positiv zu beeinflussen.
Die nachfolgenden Cross-Reference-Listen beziehen sich auf das BASIC-Programm aus Abschnitt 23 im Band 2.
Die FOSBIC Version 2 lässt die Generation entsprechender benutzerorientierter Cross-Reference-Listen noch nicht zu. Es ist geplant, sie durch die Anweisung

REFERENCES

im Compile-Modus für den Benutzer anwählbar zu machen.

```
   10   DIM A%(10,10),D(10,10),E(10,10),P(10,10)     5   MATRIZENINVERSION
   12   READ N1
   14   DATA 4
   16   PRINT "EINGABEMATRIX"
   18   FOR I=1 TO N1
   20     FOR J=1 TO N1
   22       READ D(I,J)
   24       PRINT D(I,J),
   26     NEXT J
   28     PRINT
   30   NEXT I
   32   FOR I=1 TO N1                               31   BILDEN EINER EINHEITSMATRIX
   34     FOR J=1 TO N1
   36       LET E(I,J)=0
   38       IF I<>J GOTO 42
   40       LET E(I,J)=1
1=>42     NEXT J
   44   NEXT I
   80   FOR M1=1 TO N1                              46   BEGINN DER INVERSION
   90     LET T=D(M1,M1)
  100     IF T=0 GOTO 360
  110     FOR N=1 TO N1                             105  BEGINN DES ERSTEN EINZELSCHRITTS
  120       LET E(M1,N)=E(M1,N)/T
  130       LET D(M1,N)=D(M1,N)/T
  140     NEXT N
  150     LET M=1
1=>160    IF M1=M GOTO  240                         155  BEGINN DES ZWEITEN EINZELSCHRITTS
  170     LET T=D(M,M1)
  180     FOR N=1 TO N1
  210       LET E(M,N)=E(M,N)-E(M1,N)*T
  220       LET D(M,N)=D(M,N)-D(M1,N)*T
  230     NEXT N
1=>240    LET M=M+1
  250     IF M<=N1 GOTO 160
  260   NEXT M1
  265   PRINT                                       265  ENDE DER INVERSION
  275   PRINT "INVERSE MATRIX"
  280   FOR I=1 TO N1
  282     FOR J=1 TO N1
  284       PRINT E(I,J),
  286     NEXT J
  288     PRINT
  290   NEXT I
  292   RESTORE  340
  294   FOR I=1 TO N1
  296     FOR J=1 TO N1
  298       READ D(I,J)
  300     NEXT J
  302   NEXT I
  304   FOR I=1 TO N1
  306     FOR J=1 TO N1
  308       LET P(I,J)=0
  310       FOR K=1 TO N1
  312         LET P(I,J)=P(I,J)+D(I,K)*E(K,J)
  314       NEXT K
  316     NEXT J
  318   NEXT I
  319   PRINT
  320   PRINT "KONTROLLMATRIX"
  322   FOR I=1 TO N1
  324     FOR J=1 TO N1
  326       PRINT P(I,J),
  328     NEXT J
  330     PRINT
  332   NEXT I
  340   DATA 49.96,12.23,36.78,29.43
  342   DATA 22.44,38.72,40.15,11.79
  344   DATA 36.45,19.2,37.78,6.43
  346   DATA 17.86,40.02,27.86,39.27
  350   STOP
1=>360    PRINT "EINGABEMATRIX IST SINGULAR"
  400   END
```

```
VARIABLE CROSS-REFERENCES

x---DIMENSION---x
VAR   SIZE LINE TYPE LINE   LINE   LINE   LINE   LINE   LINE   LINE   LINE   LINE
+-------------------------------------------------------------------------------+

D     10,10   10  4%    22     24     90R   130M   130R   170R   220M   220R   220
                      298    312

E     10,10   10  4%    36M    40M   120M   120R   210M   210R   210    284    312

I             4%    18     22     24     30     32     36     38     40     44
                    280    284    290    294    298    302    304    308    312
                    318    322    326    332

J             4%    20     22     24     26     34     36     38     40     42
                    282    284    286    296    298    300    306    308    312
                    316    324    326    328

K             4%   110    112    114

M             4%   150M   160    170    210    220    240M   250
M1            4%    80     90    120    130    160    170    210    220    240

N             4%   110    120    130    140    180    210    220    230
N1            4%    12     18     20     32     34     80    110    180    200
                    280    282    294    296    304    306    310    322    324

P     10,10   10  4%   308M   312M   312R   324

T             4%    90M   100    120    130    170M   210    220
+-------------------------------------------------------------------------------+

PROGRAM STATISTICS

+-------------------------------------------------------------------------------+

UNDIMENSIONED VARS :   I     J     K     M     M1    N     N1    T
+-------------------------------------------------------------------------------+

 VARIABLES REFERENCED = 11 OUT OF 93
 SIMPLE VARIABLES = 8  STRING VARIABLES = 0
 DIMENSIONED ARRAYS = 3  TOTAL REFERENCES = 115
 PROGRAM SIZE = 914 WORDS + DATA AREA 302 WORDS  RUNSIZE = 3 KB
+-------------------------------------------------------------------------------+


STATEMENT CROSS-REFERENCES

STMT     LINE   LINE   LINE   LINE   LINE   LINE   LINE   LINE   LINE   LINE   LINE
+-------------------------------------------------------------------------------+

  42<==   38T
 160<==  20UT
 240<==  160T
 360<==  1 UT

+-------------------------------------------------------------------------------+
```

5 BASIC-Anwendungsbeispiele

In Fortsetzung der im Band 1 gezeigten BASIC-Anwendungsbeispiele werden in den folgenden Abschnitten ausgewählte Probleme aus den Gebieten der Statistik, der Betriebswirtschaftslehre und der Volkswirtschaftslehre mit BASIC-Programmen gelöst, wobei der gesamte Anweisungsumfang des FOSBIC-Compilers zum Einsatz gelangt.
Die zur Problembearbeitung erforderlichen theoretischen Grundkenntnisse werden beim Leser vorausgesetzt.
Die Darstellung der Problemstruktur und des Lösungsweges beschränkt sich auf ein Minimum und ist auf den Aufbau des jeweiligen BASIC-Programmbeispiels abgestimmt. Bevorzugt wird der Einsatz der statementsparenden Matrizenanweisungen.
Gegenüber der Erstauflage (S. 180–215) sind Anweisungen, die alphanumerische Variable beinhalten, und Fileanweisungen den neuen Standards angepasst. Bei einzelnen Programmbeispielen ist neben der ursprünglichen Monolog- (Batch-)Version eine ergänzte Dialogversion abgedruckt.

51 Statistik

511 Matrizenrechnung

Die in der Programmiersprache BASIC verfügbaren Matrizenanweisungen ermöglichen dem Benutzer eine einfache und statementsparende Problemprogrammierung.
Bei der Übertragung von Matrizenoperationen in BASIC-MAT-Statements ist die exakte Einhaltung und genaue Korrespondenz der Feldgrenzen besonders zu beachten, da die Matrizengleichungen normalerweise keine Angaben über die Feldgrenzen bzw. die notwendigen Korrespondenzgrössen der Feldgrenzen enthalten.
In Fortsetzung der in Band 2 (S. 78–86 und S. 96–101) angeführten Beispiele werden zunächst einige grundlegende Matrizenoperationen dargestellt und deren Einsatzmöglichkeiten in zwei einfachen Beispielen zusammengefasst; dabei wird bei der formelmässigen Darstellung der Vereinbarung gefolgt, dass Kleinbuchstaben für Vektoren und Grossbuchstaben für Matrizen stehen.
Die Summe der Elemente eines Zeilenvektors x

$$S = \sum_{i=1}^{n} x_i = x \cdot 1^T$$

ergibt sich aus der Multiplikation mit einem Einer-Spaltenvektor.

```
10 DIM X(100), Y(100,1)
20 READ N
30 MAT READ X(N)
40 MAT  Y=CON
50 MAT  S=X*Y
60 PRINT "VEKTORSUMME =";S
70 DATA 10
75 DATA 1,2,3,4,5,6,7,8,9,10
80 END
```

```
VEKTORSUMME = 55
```

Wird ein Zeilenvektor mit seinem transponierten Vektor multipliziert, so resultiert daraus die Quadratsumme der Elemente als Skalarwert.

$$S = \sum_{i=1}^{n} x_i^2 = x \cdot x^T$$

```
10 DIM X(100), Y(100,1)
20 READ N
30 MAT READ X(N)
40 MAT  Y=TRN(X)
50 MAT  S=X*Y
60 PRINT "QUADRATSUMME =";S
70 DATA 10
80 DATA 1,2,3,4,5,6,7,8,9,10
90 END
```

```
QUADRATSUMME = 385
```

Die Produktsumme zweier Zeilenvektoren x und y

$$S = \sum_{i=1}^{n} x_i y_i$$

ergibt sich aus der Matrizenoperation

$$S = x \cdot y^T = y \cdot x^T$$

```
10 DIM X(100), Y(100), Z(100,1)
20 READ N
30 MAT READ X(N), Y(N)
35 MAT  Z=TRN(Y)
40 MAT  S=X*Z
50 PRINT "PRODUKTSUMME =";S
60 DATA 10
70 DATA 1,2,3,4,5,6,7,8,9,10
80 DATA 1,2,3,4,5,6,7,8,9,10
90 END
```

```
PRODUKTSUMME = 385
```

Für Spaltenvektoren, hier mit x und y bezeichnet, gilt

$$S = x^T \cdot y = y^T \cdot x$$

wozu obiges Programmbeispiel vom Leser modifiziert werden kann. Die Divisionssumme zweier Zeilenvektoren x und y

$$S = \sum_{i=1}^{n} \frac{x_i}{y_i}$$

verlangt zunächst die Bildung der reziproken Elementwerte des Vektors y durch die skalare Matrizenoperation

$$y' = \frac{1}{y}$$

und anschliessend die Matrizenoperation

$$S = x \cdot y'^T.$$

Eine Transponierung des Vektors y' entfällt, wenn y' bereits als Spaltenvektor vorgegeben ist.

```
10 DIM X(100),  Y(100,1)
20 READ N
30 MAT   X=ZER(N)
40 MAT   Y=ZER(N,1)
50 MAT READ X, Y
60 MAT   Y=(1)/Y
70 MAT   S=X*Y
80 DATA 10
82 DATA 1,2,3,4,5,6,7,8,9,10
84 DATA 1,2,3,4,5,6,7,8,9,10
90 PRINT "DIVISIONSSUMME =";S
99 END
```

```
DIVISIONSSUMME = 10
```

Multipliziert man einen Zeilenvektor, dessen Elemente alle auf Eins gesetzt sind mit einer Matrix, so enthält der Ergebnisvektor die Spaltensummen der Matrix.

$$c = 1 \cdot X$$

Umgekehrt ergeben sich aus der Matrizenoperation

$$c^T = X \cdot 1^T$$

die Zeilensummen als Elementwerte des Spaltenvektors c^T.

```
 10 REM SUMMIERENDE VEKTOREN S UND T
 20 REM SPALTENSUMME = S * MATRIX
 30 REM ZEILENSUMME = MATRIX * T
 40 DIM A(3,4), B(1,4), C(3,1), S(1,3), T(4,1)
 50 MAT INPUT A
 60 MAT  S = (1)
 70 MAT  T = (1)
 80 MAT  B = S*A
 90 MAT  C = A*T
100 MAT PRINT A, B
110 MAT PRINT C
120 END LIST DATA


LIST OF DATA AFTER END STATEMENT


2, 4, 0, 1, 3, 4, 1, 1, 5, 0, 2, 1
*


MATRIX *A* ZEILE *   1*
   2                 4                 0                 1
MATRIX *A* ZEILE *   2*
   3                 4                 1                 1
MATRIX *A* ZEILE *   3*
   5                 0                 2                 1
MATRIX *B* ZEILE *   1*
  10                 8                 3                 3
MATRIX *C* ZEILE *   1*
   7
MATRIX *C* ZEILE *   2*
   9
MATRIX *C* ZEILE *   3*
   8
```

Zur Summenbildung aller Elemente einer Matrix

$$S = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$$

werden die Matrizenoperationen

$$z^T = X \cdot 1^T$$

und

$$S = 1 \cdot z^T$$

angewendet.

```
 10 DIM X(10,10),Y(10,1),Z(10,1)
 20 READ M, N
 25 MAT  X=ZER(M,N)
 30 MAT READ X
 40 MAT  Y=CON(N,1)
 45 MAT  Z=ZER(M,1)
 50 MAT  Z=X*Y
 60 MAT  Y=CON(1,M)
 70 MAT  S=Y*Z
 80 PRINT "MATRIZENSUMME =";S
 82 DATA 3,4
 84 DATA 1,2,3,4,5,6,7,8,9,10,11,12
 90 END

MATRIZENSUMME =  78
```

Ebenso einfach lassen sich das Zeilen- und Spaltenprodukt eines Vektors b mit einer Matrix A

$$c_i = \sum_{j=1}^{n} b_i a_{ij}$$

$$c_j = \sum_{i=1}^{m} a_{ij} b_j$$

durch die Matrizenoperationen

$$c = b \cdot A$$

$$c^T = A \cdot b^T$$

bilden, wobei c^T und b^T Spaltenvektoren darstellen.

```
10 DIM A(10,1),B(10,10),C(1,10)
20 READ N,M
30 MAT  A=ZER(N,1)
40 MAT  B=ZER(M,N)
50 MAT  C=ZER(M,1)
60 MAT READ A,B
70 MAT  C=B*A
80 MAT PRINT C
90 DATA 3,4
92 DATA 1,2,3
94 DATA 1,-2,4, 3,2,1,  0.0,-4,-2,  2,5,-1
99 END
```

```
MATRIX *C* ZEILE *   1*
    9
MATRIX *C* ZEILE *   2*
   10
MATRIX *C* ZEILE *   3*
  -14
MATRIX *C* ZEILE *   4*
    9
```

```
10 DIM A(10,1),B(10,10),C(1,10)
20 READ N,M
30 MAT  A=ZER(1,M)
40 MAT  B=ZER(M,N)
50 MAT  C=ZER(1,N)
60 MAT READ A,B
70 MAT  C=A*B
80 MAT PRINT C
90 DATA 3,4
92 DATA 1,2,3,4
94 DATA 1,-2,4, 3,2,1,  0.0,-4,-2,  2,5,-1
99 END
```

```
MATRIX *C* ZEILE *   1*
   15            10            -4
```

Statistische Masszahlen[1] wie Mittelwert, Varianz und Standardabweichung lassen sich mit Matrizenanweisungen in einem kurzen Programm bestimmen, wobei die Matrizenoperationen

$$X2 = x \cdot x^T$$

1 Vgl. Bd. 1, Abs. 431.

die Quadratsumme und

$$X1 = x \cdot 1^T$$

die Summe der Beobachtungswerte liefern.

```
10 DIM X(100),Z(100,1)
20 READ N
30 MAT READ X(N)
40 MAT  Z=TRN(X)
50 MAT  X2=X*Z
60 MAT  Z=CON
70 MAT  X1=X*Z
80 DATA 17
85 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
90 PRINT "      ANZAHL","      SUMME","   QUADRATSUMME","   MITTELWERT",
92 PRINT "      VARIANZ","STANDARDABWEICHUNG"
94 PRINT N,X1,X2,X1/N,(N*X2-X1**2)/(N*(N-1)),SQR((N*X2-X1**2)/(N*(N-1)))
95 END
```

```
 ANZAHL     SUMME     QUADRATSUMME     MITTELWERT        VARIANZ     STANDARDABWEICHUNG
   17        153         1785              9              25.5            5.04975
```

Der Chi-Quadrat-Test für m Merkmale und n Stichproben wird nach der Formel

$$Chi = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{(x_{ij} - e_{ij})^2}{e_{ij}}$$

durchgeführt, wobei sich die Erwartungswerte e_{ij} aus der Beziehung

$$e_{ij} = \frac{T_i \cdot S_j}{X1}$$

errechnen. T_i ergibt sich aus der Summe der absoluten Häufigkeiten x_{ij} über alle Stichproben eines Merkmals

$$T_i = \sum_{j=1}^{n} x_{ij} \qquad i = 1(1)m$$

und S_j als Summe aller Merkmalshäufigkeiten innerhalb einer Stichprobe

$$S_j = \sum_{i=1}^{m} x_{ij} \qquad j = 1(1)n$$

X1 ist die Summe aller Häufigkeiten.

$$X1 = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$$

Die Erwartungswertmatrix folgt aus den Matrizenoperationen

$$\begin{aligned} T &= X \cdot 1^T \\ S &= 1 \cdot X \\ X1 &= 1 \cdot T \\ E' &= T \cdot S \\ \text{und } E &= E'/(X1) \end{aligned}$$

wobei T ein Spaltenvektor mit den Feldgrenzen (M,1), S ein Zeilenvektor mit den Feldgrenzen (1,N), X und E Matrizen mit den Feldgrenzen (M,N) und X1 ein Skalar sind. Die Berechnung des Chi-Quadrat-Wertes muss mit FOR/NEXT-Schleifen erfolgen, da für derartige Summenbildung keine Matrizenanweisung existiert[2]. Um die Wahrscheinlichkeit des Chi-Quadrat-Tests aus den entsprechenden Tabellen ablesen zu können, wird der Freiheitsgrad

$$f = (M-1)(N-1)$$

enötigt.

```
1000 DIM X(10,10),E(10,10),Z(10),S(10,1),T(10,1)
1005 READ M,N
1010 IF M<0 GOTO  1175
1015 MAT X=ZER(M,N)
1020 MAT READ X(M,N)
1025 PRINT "HAEUFIGKEITSTABELLE"
1030 PRINT "--------------------"
1035 MAT PRINT X(M,N)
1040 MAT S=CON(N,1)
1045 MAT T=ZER(M,1)
1050 MAT T=X*S
1055 MAT Z=CON(M)
1060 MAT X1=Z*T
1065 MAT S=ZER(1,N)
1070 MAT S=Z*X
1075 MAT E=ZER(M,N)
1080 MAT E=T*S
1085 MAT E=E/(X1)
1090 PRINT
1095 PRINT "ERWARTUNGSWERTTABELLE"
1100 PRINT "---------------------"
1105 MAT PRINT E(M,N)
1110 FOR I=1 TO M
1115  FOR J=1 TO N
1120   LET X(I,J)=((X(I,J)-E(I,J))**2)/E(I,J)
1125   LET S1=S1+X(I,J)
1130  NEXT J
1135 NEXT I
1140 PRINT
1145 PRINT "CHI-WERTTABELLE"
1150 PRINT "---------------"
1155 MAT PRINT X(M,N)
1160 PRINT
1165 PRINT "CHI-QUADRATWERT =";S1,"BEI **";(M-1)*(N-1);"* FREIHEITSGRADEN"
1170 GOTO  1005
1175 STOP
1180 DATA 8, 2
1185 DATA 4,5,  8,6,  12,15,  21,20,  45,43,  32,29,  28,28,  7,9
1190 DATA -1,-1
1195 END
```

2 Vgl. Weber und Türschmann, FOSBIC-Compiler, Abs. 351.21.

```
HAEUFIGKEITSTABELLE
-------------------
        4              5
        8              6
       12             15
       21             20
       45             43
       32             29
       28             28
        7              9

ERWARTUNGSWERTTABELLE
---------------------
        4.52885        4.47115
        7.04487        6.95513
       13.5865        13.4135
       20.6314        20.3686
       44.2821        43.7179
       30.6955        30.3045
       28.1795        27.8205
        8.05128        7.94872

CHI-WERTTABELLE
---------------
      0.0617549      0.0625517
       0.129494       0.131165
       0.185265       0.187655
     0.00658503        0.00667
      0.0116402      0.0117904
      0.0554376       0.056153
     0.00114323     0.00115798
       0.137269       0.139041

CHI-QUADRATWERT = 1.18477      BEI * 7 * FREIHEITSGRADEN
```

5512 Input-Output Analyse

Das Input-Output Modell – Abbildung 29 –, auch Transaktionsmatrix genannt[3], beschreibt die mengenmässigen Beziehungen zwischen Aktivitäten innerhalb eines Systems, wobei statisch offene, statisch geschlossene und dynamische Modelle[4] diskutiert werden. Hier soll der Fall des statisch offenen Modells betrachtet werden.

In Abbildung 29 stellen a_{ij} die Zwischentransaktionen von i nach j dar. b_{ik} sind die Endbeträge des Outputs und c_{kj} die primären Inputbeträge. In Matrizenschreibweise ergibt sich die Transaktionsmatrix T

$$T = \left[\begin{array}{c|c|c} A & B & y \\ \hline C & O & o \end{array}\right]$$

3 Shawki M. Farag. Input-Output Analysis: Applications to Business Accounting. Urbana 1967, p. 32.

4 Farag, Input-Output Analysis, p. 52–57. Heinz Stöwe und Erich Härtter. Lehrbuch der Mathematik für Volks- und Betriebswirte. 2. Aufl., Göttingen 1972, S. 157.

mit den Feldgrenzen (M,M) für A, (M,N) für B und (O,M) für C. O ist eine Nullmatrix und y der Spaltenvektor des gesamten Outputs. Normalerweise wird die Endbetragsmatrix B durch Spaltensummation

$$B_i = \sum_{l=1}^{N} \bar{b}_{il} \qquad i = 1(1)M$$

zu einem Spaltenvektor b und die Matrix C durch Zeilenaddition

$$c_j = \sum_{k=1}^{O} \bar{c}_{kj} \qquad j = 1(1)M$$

zu einem Zeilenvektor c zusammengefasst. Die Transaktionsmatrix T erhält damit die Form

$$T = \left[\begin{array}{c|c|c} A & b & y \\ \hline c & o & o \end{array}\right]$$

Bei dem statisch offenen Input-Output Modell wird die Frage gestellt, wie die Inputgrössen a_{ij} und $\bar{c}_{kj}$ für exogen vorgegebene Endbeträge b_{il} zu berechnen sind[5].
Léon Walras[6] und Wassily Leontief[7] begründeten das Konzept, wonach die technischen Koeffizienten (Produktionskoeffizienten) a_{ij} aus der Zwischentransaktionsmatrix A berechnet werden.

$$a'_{ij} = \frac{a_{ij}}{y_j} \qquad \begin{array}{l} i = 1(1)M \\ j = 1(1)M \end{array}$$

Für die Elemente der primären Inputmatrix C gilt eine analoge Umformung.

$$c'_{kj} = \frac{c_{kj}}{y_j} \qquad \begin{array}{l} j = 1(1)M \\ i = 1(1)O \end{array}$$

Die technischen Koeffizienten a'_{ij} und c'_{kj} drücken jeweils den Anteil aus, den der Input i bzw. k zur Erstellung einer Outputmenge beiträgt. Die Transaktionsmatrix lautet damit:

5 Richard Mattessich. Die wissenschaftlichen Grundlagen des Rechnungswesens. Düsseldorf 1970, S. 222.
6 Mattessich, Die wissenschaftlichen Grundlagen des Rechnungswesens, S. 215–222; Farag, Input-Output Analysis, p. 28–38.
7 Stöwe und Härtter, Lehrbuch der Mathematik für Volks- und Betriebswirte, S. 157.

Input \ Output		Zwischentransaktionen 1	2	... j	... M	Endbeträge 1	2	... 1	...N	Gesamter Output Zeilensummen
	1	a_{11}	a_{12}	... a_{1j}	... a_{1M}	b_{11}	b_{12}	... b_{1i}	... b_{1N}	$\sum_{j=1}^{M} a_{1j} + \sum_{i=1}^{N} b_{1i}$
	2	a_{21}	a_{22}	a_{2j}	a_{2M}	b_{21}	b_{22}	b_{2i}	b_{2N}	$\sum_{j=1}^{M} a_{2j} + \sum_{i=1}^{N} b_{2i}$
	i	a_{i1}	a_{i2}	a_{ij}	a_{iM}	b_{i1}	b_{i2}	b_{ii}	b_{iN}	$\sum_{j=1}^{M} a_{ij} + \sum_{i=1}^{N} b_{ii}$
	M	a_{M1}	a_{M2}	... a_{Mj}	... a_{MM}	b_{M1}	b_{M2}	... b_{Mi}	... b_{MN}	$\sum_{j=1}^{M} a_{Mj} + \sum_{i=1}^{N} b_{Mi}$
primärer Input	1	c_{11}	c_{12}	... c_{1j}	... c_{1M}					
	2	c_{21}	c_{22}	c_{2j}	c_{2M}					Summen des primären Inputs
	k	c_{k1}	c_{k2}	c_{kj}	c_{kM}					$\sum_{j=1}^{M} c_{kj}$
	O	c_{O1}	c_{O2}	... c_{Oj}	... c_{OM}					
Spaltensummen		$\sum_{i=1}^{M} a_{i1}$ / $\sum_{k=1}^{O} c_{k1}$	$\sum_{i=1}^{M} a_{i2}$ / $\sum_{k=1}^{O} c_{k2}$	$\sum_{i=1}^{M} a_{ij}$ / $\sum_{k=1}^{O} c_{kj}$	$\sum_{i=1}^{M} a_{iM}$ / $\sum_{k=1}^{O} c_{kM}$	Summen der Endbetr. $\sum_{i=1}^{N} b_{ii}$				$\sum_{j=1}^{M} (\sum_{i=1}^{M} a_{ij} + \sum_{i=1}^{N} b_{ii})$ oder $\sum_{j=1}^{M} (\sum_{i=1}^{M} a_{ij} + \sum_{k=1}^{O} c_{kj})$

Abbildung 29. Input-Output Tabelle

$$T = \left[\begin{array}{c|c|c} A' & B & y \\ \hline C' & O & o \end{array}\right] \text{ oder } T = \left[\begin{array}{c|c|c} A' & b & y \\ \hline C' & o & o \end{array}\right]$$

Wird der obere Teil der Transaktionsmatrix als Matrizengleichung

$$A'y + b = y$$

geschrieben, so ergibt sich durch Umformen und Einführen einer Einheitsmatrix E die Gleichung

$$(E - A')y = b.$$

Durch Multiplikation mit der Inversen $(E - A')^{-1}$ findet sich die Lösungsgleichung für den Outputvektor y,

$$y = (E - A')^{-1}b$$

wobei sich der Endbetragsvektor b wiederum durch Spaltenaddition aus einer Endbetragsmatrix B ergeben kann.

Die Elemente a'_{ij} und c'_{kj} der neuen Inputmatrizen A″ und C″ ergeben sich aus den Matrizenmultiplikationen

$$A'' = A' \cdot P \quad \text{und} \quad C'' = C' \cdot P,$$

wobei P eine Matrix mit den Feldgrenzen (M,M) ist, deren Diagonale den berechneten Outputvektor y enthält. Bei der Ermittlung der technischen Koeffizienten nimmt die Matrix P in ihrer Diagonalen die reziproken Elementwerte $1/y_j$ des Outputs im Starttableau auf.

$$P = \begin{bmatrix} \frac{1}{y_1} & 0 & 0 \ldots 0 \\ 0 & \frac{1}{y_2} & 0 \quad 0 \\ \vdots & & \vdots \\ 0 & 0 & 0 \quad \frac{1}{y_M} \end{bmatrix}$$

Hierdurch lässt sich die Berechnung der technischen Koeffizienten auf die Matrizenoperationen

$$A' = A \cdot P \quad \text{und} \quad C' = C \cdot P$$

zurückführen.

Das folgende Programmbeispiel zeigt, dass die eigentliche Input-Output Analyse nur wenige MAT-Statements beansprucht, während zwei Drittel der Programmanweisungen zum Löschen und Redimensionieren der Felder und zur Ausgabe der Input-Output Tabellen benötigt werden. Das BASIC-Programm berücksichtigt eine vorgebbare Beschriftung der Tabellen durch Texteingabe, wobei je Textzeile maximal 10 Zeichen einzugeben sind.

```
1000 DIM A(10,10),B(10,10),C(10,10),P(10,10),Z(10,1),Y(10,1),S(10,1),E(10,10)
1005 DIM D(10,10),F$(10)*10,G$(10)*10,H$(10)*10,L(10,1),Q$*50,K(10)
1010 DIM F(10,2),G(10,2),H(10,2)
1015 READ Q$
1020 IF Q$="ENDE" GOTO 1635
1025 READ M,N,O
1030 MAT A=ZER(M,M)
1035 MAT B=ZER(M,N)
1040 MAT C=ZER(O,M)
1045 MAT Z=ZER(M,1)
1050 MAT Y=ZER(M,1)
1055 MAT P=ZER(M,M)
1060 MAT D=ZER(M,M)
1065 MAT L=ZER(O,1)
1070 MAT F=ZER(M,2)
1075 MAT G=ZER(N,2)
1080 MAT H=ZER(O,2)
1085 MAT READ A,B,C
1090 MAT READ F$(M),G$(N),H$(O)
```

```
1095 MAT S=CON(M,1)
1100 MAT Z=A*S
1105 MAT L=C*S
1110 MAT S=CON(N,1)
1115 MAT Y=B*S
1120 MAT Y=Z+Y
1125 MAT S=CON(1,M)
1130 MAT K=ZER(N)
1135 MAT K=S*B
1140 LET N9=1
1145 GOTO  1260
1150 FOR I=1 TO M
1155  LET P(I,I)=1/Y(I,1)
1160 NEXT I
1165 MAT A=A*P
1170 MAT C=C*P
1175 MAT S=CON(M,1)
1180 MAT L=C*S
1185 IF N9=2 GOTO  1260
1190 MAT E=IDN(M,M)
1195 MAT D=E-A
1200 MAT D=INV(D)
1205 MAT READ B
1210 MAT S=CON(N,1)
1215 MAT Y=B*S
1220 MAT Y=D*Y
1225 MAT S=CON(1,M)
1230 MAT K=S*B
1235 LET N9=2
1240 FOR I=1 TO M
1245  LET P(I,I)=Y(I,1)
1250 NEXT I
1255 GOTO  1165
1260 IF N9=2 GOTO  1290
1265 PRINT TAB (46);"STARTTABLEAU"
1270 PRINT TAB (46);"------------"
1275 PRINT
1280 PRINT Q$
1285 GOTO  1315
1290 PAGE
1295 PRINT TAB (47);"ENDTABLEAU"
1300 PRINT TAB (47);"----------"
1305 PRINT
1310 PRINT Q$
1315 PRINT
1320 PRINT " "," "," ","----OUTPUT----"
1325 PRINT
1330 PRINT "----INPUT----",
1335 FOR I=1 TO M
1340  PRINT "      ";P$(I),
1345 NEXT I
1350 FOR I=1 TO N
1355  PRINT "      ";G$(I),
1360 NEXT I
1365 PRINT "      TOTAL"
1370 FOR I=1 TO M
1375  PRINT P$(I),
1380  FOR J=1 TO M
1385   PRINT A(I,J),
1390  NEXT J
1395  FOR J=1 TO N
1400   PRINT B(I,J),
1405  NEXT J
1410  PRINT Y(I,1)
1415 NEXT I
1420 FOR I=1 TO O
1425  PRINT H$(I),
1430  FOR J=1 TO M
1435   PRINT C(I,J),
1440  NEXT J
1445  FOR K=1 TO N
1450   PRINT " ",
1455  NEXT K
1460  PRINT L(I,1)
1465 NEXT I
1470 PRINT
```

```
1475 PRINT "TOTAL",
1480 LET T1=0
1485 FOR J=1 TO M
1490  PRINT Y(J,1),
1495  LET T1=T1+Y(J,1)
1500 NEXT J
1505 FOR J=1 TO N
1510  PRINT K(J),
1515  LET T1=T1+K(J)
1520 NEXT J
1525 PRINT T1
1530 IF N9=1 GOTO  1150
1535 DATA "--------------- STATISTIK BEISPIEL ---------------"
1540 DATA 3,2,4
1545 DATA 11,7,8
1550 DATA 10,13,6
1555 DATA 25,8,17
1560 DATA 100,80
1565 DATA 70,120
1570 DATA 100,150
1575 DATA 30,43,60
1580 DATA 40,30,80
1585 DATA 70,60,50
1590 DATA 20,58,79
1595 DATA "TEXT-1","TEXT-2","TEXT-3"
1600 DATA "OUTPUT-4","OUTPUT-5"
1605 DATA "INPUT-1","INPUT-2","INPUT-3","INPUT-4"
1610 DATA 170,350
1615 DATA 190,90
1620 DATA 160,100
1625 PAGE
1630 GOTO  1015
1635 STOP
1640 END LIST DATA
```

513 Multiple lineare Regression

Wird zwischen r unabhängigen Zufallsvariablen x_r und einer abhängigen Zufallsvariablen y ein linarer Zusammenhang vermutet, der sich durch die Gleichung

$$y = b_o + b_1x_1 + b_2x_2 + \ldots + b_rx_r = b_o + \sum_{p=1}^{r} b_px_p$$

ausdrückt, so kann dies durch eine multiple lineare Regressionsrechnung nachgewiesen werden.[8]
Nach der Methode der kleinsten Quadrate ergibt sich der Ansatz

$$\sum_{t=1}^{n}(y_t - (b_o + b_1x_{t1} + \ldots + b_rx_{tr}))^2 \stackrel{!}{=} \min$$

8 Vgl. Bd. 1, Abs. 432.

STARTTABLEAU

---------- STATISTIK BEISPIEL ----------

----INPUT----	TEXT-1	TEXT-2	TEXT-3	----OUTPUT---- OUTPUT-4	OUTPUT-5	TOTAL
TEXT-1	11	7	8	100	80	206
TEXT-2	10	13	6	70	120	219
TEXT-3	25	8	17	100	150	300
INPUT-1	30	43	60			133
INPUT-2	40	30	80			150
INPUT-3	70	60	50			180
INPUT-4	20	58	79			157
TOTAL	206	219	300	270	350	1345

ENDTABLEAU

---------- STATISTIK BEISPIEL ----------

----INPUT----	TEXT-1	TEXT-2	TEXT-3	----OUTPUT---- OUTPUT-4	OUTPUT-5	TOTAL
TEXT-1	30.4816	10.7022	9.65392	170	350	570.838
TEXT-2	27.7106	19.8755	7.24044	190	90	334.827
TEXT-3	69.2764	12.2311	20.5146	160	100	362.022
INPUT-1	83.1317	65.7422	72.4044			221.278
INPUT-2	110.842	45.8667	96.5392			253.248
INPUT-3	193.974	91.7333	60.337			346.044
INPUT-4	55.4211	88.6755	95.3325			239.429
TOTAL	570.838	334.827	362.022	520	540	2327.69

wobei x_{ti} für die t Beobachtungswerte der Zufallsvariablen i steht. Wird die unabhängige Zufallsvariable $x_{to} = 1$ $(t = 1(1)n)$ eingeführt[9], so lautet der Ansatz

$$\sum_{t=1}^{n}(y_t - b_o x_{to} - b_1 x_{t1} - \ldots - b_r x_{tr})^2 \stackrel{!}{=} \min.$$

Nach partieller Differentiation und Nullsetzung der r + 1 Partialableitungen ergibt sich ein Normalgleichungssystem mit r + 1 Gleichungen für die gesuchten Regressionsparameter b_p $(p = 0(1)r)$.

$$\begin{aligned}
&\sum_{t=1}^{n} y_t x_{to} - b_o \sum_{t=1}^{n} x_{to}^2 - b_1 \sum_{t=1}^{n} x_{t1} x_{to} - \ldots - b_n \sum_{t=1}^{n} x_{tn} x_{to} = 0 \\
&\sum_{t=1}^{n} y_t x_{t1} - b_o \sum_{t=1}^{n} x_{to} x_{t1} - b_1 \sum_{t=1}^{n} x_{t1}^2 - \ldots - b_n \sum_{t=1}^{n} x_{tn} x_{t1} = 0 \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&\sum_{t=1}^{n} y_t x_{tp} - b_o \sum_{t=1}^{n} x_{to} x_{tp} - b_1 \sum_{t=1}^{n} x_{t1} x_{tp} - \ldots - b_n \sum_{t=1}^{n} x_{tn} x_{tp} = 0 \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&\sum_{t=1}^{n} y_t x_{tr+1} - b_o \sum_{t=1}^{n} x_{to} x_{tr+1} - \qquad\qquad \ldots - b_n \sum_{t=1}^{n} x_{tn} x_{tr+1} = 0
\end{aligned}$$

In Matrizenschreibweise stellt sich das Gleichungssystem in der Form

$$X'y - (X'X)b = 0$$

dar, wobei X′ die transponierte Eingabematrix

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1r} \\ 1 & x_{21} & x_{22} & \ldots & x_{2r} \\ \vdots & & & & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nr} \end{bmatrix}$$

der r unabhängigen Variablen mit n Beobachtungen und den Feldgrenzen (N,R + 1) ist. Der Spaltenvektor y

9 Lucy J. Slater. FORTRAN Programs for Economists. Occasional Paper 13, Cambridge 1967, p. 76.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

mit den Feldgrenzen (N,1) enthält die n Beobachtungen der abhängigen Variablen, der Spaltenvektor b

$$b = \begin{bmatrix} b_o \\ b_1 \\ b_2 \\ \vdots \\ b_r \end{bmatrix}$$

mit den Feldgrenzen (R + 1,1) die Regressionskoeffizienten für die r-fache multiple Regression.
Durch die Anwendung der Matrizenschreibweise lässt sich das Lösungsverfahren und die Berechnung der statistischen Werte in einfacher und übersichtlicher Weise beschreiben[10].
Wird die Matrizengleichung

$$X'y - (X'X)\,b = 0$$

umgeformt in

$$(X'X)\,b = X'y$$

und durch Multiplikation mit der Inversen $(X'X)^{-1}$ nach b aufgelöst, so gilt

$$b = (X'X)^{-1} \cdot X'y$$

X'X ist die Kreuzproduktmatrix der unabhängigen Variablen mit den Feldgrenzen (R + 1,R + 1) und X'y der Kreuzproduktspaltenvektor zwischen abhängigen und unabhängigen Variablen mit den Feldgrenzen (R + 1,1).
Die Summe S1 und die Quadratsumme S2 der abhängigen Beobachtungswerte erhält man aus den Matrizenoperationen

$$S1 = 1' \cdot y$$

$$S2 = y \cdot y'$$

10 John Neter and William Wassermann. Applied Linear Statistical Models. Homewood 1974, p. 225–233.

Die Varianz von y ermittelt sich aus

$$S5 = S2 - \frac{S1^2}{N}$$

Das Matrizenprodukt

$$S3 = b'X'y$$

ist die Basis zur Berechnung von:

- Summe der Abweichungsquadrate innerhalb der Regression

$$S7 = S3 - \frac{S1^2}{N}$$

- mittlere Summe der Abweichungsquadrate innerhalb der Regression

$$S7M = \frac{S7}{R - 1}$$

- Summe der Abweichungsquadrate um die Steigung der Regression

$$S8 = S2 - S3$$

- mittlere Summe der Abweichungsquadrate um die Steigung der Regression

$$S4 = \frac{S8}{N - R}$$

- F-Wert

$$F = \frac{S7}{S8} \cdot \frac{N - R}{R - 1}$$

- Standardfehler der Schätzung

$$SF = \sqrt{S4}$$

- multiplen Bestimmtheitsmassen

$$B1 = \frac{S7}{S5} \text{ oder } B2 = 1 - \frac{S8}{S5} \cdot \frac{N - 1}{N - R}$$

- multiplen Korrelationskoeffizienten

$$K1 = \sqrt{B1} \text{ oder } K2 = \sqrt{B2}$$

Multipliziert man die Elemente der inversen Kreuzproduktmatrix $H = (X'X)^{-1}$ mit dem Skalarwert der mittleren Summe der Abweichungsquadrate um die Steigung der Regression, so ergibt sich daraus die Varianz-Covarianz-Matrix H^+.

$$H^+ = H \cdot (S4)$$

Der Standardfehler der Regressionskoeffizienten errechnet sich aus den positiven Wurzeln der Hauptdiagonalelemente der Varianz-Covarianz Matrix.

$$z_i = \sqrt{h_{ii}} \qquad i = 1(1)R$$

Die Werte des t-Tests ergeben sich durch Division des Regressionskoeffizienten mit dem jeweiligen Standardfehler.

$$t_i = \frac{b_i}{z_i}$$

Die Differenz zwischen beobachteten und durch die Regression geschätzten y-Werten

$$l = y - z,$$

liefert den Residuenvektor l mit den gleichen Feldgrenzen, wobei sich der Spaltenvektor z mit den Feldgrenzen (N,1) aus der Matrizenmultiplikation

$$z = X \cdot b$$

ergibt.
Als statistisches Mass einer Autokorrelation werden der Durbin-Watson-Wert

$$d = \frac{\sum_{i=2}^{N}(l_i - l_{i-1})^2}{\sum_{i=1}^{N} l_i^2}$$

und das von Neuman-Verhältnis

$$v = d \cdot \frac{N}{N-1}$$

angegeben.
Im folgenden BASIC-Programm sind so weitgehend wie möglich diesel-

ben Bezeichnungen für Matrizen, Vektoren und statistische Werte wie oben gewählt. Besondere Aufmerksamkeit sollte in den einzelnen Matrizenoperationen der Korrespondenz der Feldgrenzen gelten.

```
1000 DIM X(15,10),Y(15,1),L(15,1),H(15,10),Z(15,10),B(15,1)
1005 READ N,R
1010 LET R=R+1
1015 IF N-R<=0 GOTO  1540
1020 MAT X=ZER(N,R)
1025 MAT Z=ZER(R,N)
1030 MAT READ X
1035 PRINT "EINGABEWERTE DER UNABHAENGIGEN VARIABLEN"
1040 PRINT "----------------------------------------"
1045 PRINT " ",
1050 FOR I=2 TO R
1055  PRINT TAB (3+15*(I-1));"X(";I-1;")";
1060 NEXT I
1065 PRINT
1070 MAT PRINT X
1075 MAT Z=TRN(X)
1080 MAT H=ZER(R,R)
1085 MAT H=Z*X
1090 PRINT
1095 PRINT "KREUZPRODUKTMATRIX DER UNABHAENGIGEN VARIABLEN"
1100 PRINT "---------------------------------------------"
1105 MAT PRINT H
1110 MAT H=INV(H)
1115 MAT Y=ZER(N,1)
1120 MAT READ Y
1125 PRINT
1130 PRINT "EINGABEWERTE DER ABHAENGIGEN VARIABLEN Y"
1135 PRINT "----------------------------------------"
1140 MAT PRINT Y
1145 MAT L=CON(1,N)
1150 MAT S1=L*Y
1155 MAT L=TRN(Y)
1160 MAT S2=L*Y
1165 MAT L=ZER(R,1)
1170 MAT L=Z*Y
1175 PRINT
1180 PRINT "KREUZPRODUKTVEKTOR DER ABHAENGIGEN UND UNABHAENGIGEN VARIABLEN"
1185 PRINT "-------------------------------------------------------------"
1190 MAT PRINT L
1195 MAT B=ZER(R,1)
1200 MAT B=H*L
1205 PRINT
1210 PRINT "REGRESSIONSKOEFFIZIENTEN"
1215 PRINT "------------------------"
1220 MAT PRINT B
1225 MAT Z=ZER(1,R)
1230 MAT Z=TRN(B)
1235 MAT S3=Z*L
1240 LET S4=(S2-S3)/(N-R)
1245 LET S5=S2-(S1**2)/N
1250 LET S7=S3-(S1**2)/N
1255 LET S8=S2-S3
1260 MAT H=H*(S4)
1265 PAGE
1270 PRINT "STATISTISCHE WERTE ZUR MULTIPLEN REGRESSION"
1275 PRINT "-------------------------------------------"
1280 PRINT
1285 PRINT "VARIANZ-COVARIANZ MATRIX"
1290 PRINT "------------------------"
1295 MAT PRINT H
1300 PRINT
1305 PRINT "SUMME Y       =";S1
1310 PRINT "QUADRATSUMME  =";S2
1315 PRINT "VARIANZ VON Y =";S5
1320 PRINT
1325 PRINT "SUMME DER","MITTLERE SUMME"
1330 PRINT "ABWEICHUNGS-","ABWEICHUNGS-"
1335 PRINT "QUADRATE","QUADRATE","FREIHEITSGRADE","   URSACHE"
1340 PRINT S7,S7/(R-1),R-1,"INNERHALB DER REGRESSION"
```

```
111345 PRINT S8,S4,N-R,"UM DIE REGRESSIONSSTEIGUNG"
111350 PRINT
111355 PRINT "F-WERT =";S7/S8*(N-R)/(R-1)
111360 PRINT "STANDARDFEHLER DER SCHAETZUNG =";SQR(S4)
111365 PRINT "MULTIPLE BESTIMMTHEITSMASSE        =";S7/S5, TAB (50);"ODER =";
111370 PRINT 1-S8/S5*(N-1)/(N-R)
111375PRINT"MULTIPLE KORRELATIONSKOEFFIZIENTEN =";SQR(S7/S5),TAB(50);"ODER =";
111380 PRINT SQR(1-S8/S5*(N-1)/(N-R))
111385 PRINT
111390 MAT Z=ZER(N,1)
111395 PRINT "T-TEST WERTE"
111400 PRINT "------------"
111405 FOR I=1 TO R
111410  LET Z(I,1)=SQR(H(I,I))
111415  PRINT B(I,1)/Z(I,1)
111420 NEXT I
111425 PRINT
111430 PRINT "STANDARDFEHLER DER REGRESSIONSKOEFFIZIENTEN"
111435 PRINT "-----------------------------------------"
111440 MAT PRINT Z(R,1)
111445 PRINT
111450 MAT Z=X*B
111455 MAT L=ZER(N,1)
111460 MAT L=Y-Z
111465 FOR J=1 TO N
111470  LET D=D+L(J,1)**2
111475  IF J=1 GOTO  1485
111480  LET D1=D1+(L(J,1)-L(J-1,1))**2
111485 NEXT J
111490 PRINT "BEOBACHTUNGSWERTE","GESCHAETZTE Y-WERTE","     RESIDUEN"
111495 PRINT "-----------------","--------------------","     --------"
111500 FOR J=1 TO N
111505  PRINT Y(J,1),"  ",Z(J,1),"  ",L(J,1)
111510 NEXT J
111515 PRINT
111520 PRINT "DURBIN-WATSON TEST =";D1/D
111525 PRINT
111530 PRINT "VON NEUMANN VERHAELTNIS =";D1/D*(N/(N-1))
111535 STOP
111540PRINT"BEI EINER *";R-1;"-FACHEN MULTIPLEN REGRESSION MUSS DIE ANZAHL DER";
111545 PRINT "STICHPROBEN GROESSER ALS *";R+1;"* SEIN"
111550 STOP
111555 DATA 13,4
111560 DATA 1,7,26,6,60
111565 DATA 1,1,29,15,52
111570 DATA 1,11,56,8,20
111575 DATA 1,11,31,8,47
111580 DATA 1,7,52,6,33
111585 DATA 1,11,55,9,22
111590 DATA 1,3,71,17,6
111595 DATA 1,1,31,22,44
111600 DATA 1,2,54,18,22
111605 DATA 1,21,47,4,26
111610 DATA 1,1,40,23,34
111615 DATA 1,11,66,9,12
111620 DATA 1,10,68,8,12
111625DATA 78.5,74.3,104.3,87.6,95.9,109.2,102.7,72.5,93.1,115.9,83.8,113.3,109.4
111630 END
```

```
EINGABEWERTE DER UNABHAENGIGEN VARIABLEN
----------------------------------------
              X( 1 )        X( 2 )        X( 3 )        X( 4 )
        1           7            26             6            60
        1           1            29            15            52
        1          11            56             8            20
        1          11            31             8            47
        1           7            52             6            33
        1          11            55             9            22
        1           3            71            17             6
        1           1            31            22            44
        1           2            54            18            22
        1          21            47             4            26
        1           1            40            23            34
        1          11            66             9            12
        1          10            68             8            12
```

KREUZPRODUKTMATRIX DER UNABHAENGIGEN VARIABLEN

13	97	626	153	390
97	1139	4922	769	2620
626	4922	33050	7201	15739
153	769	7201	2293	4628
390	2620	15739	4628	15062

EINGABEWERTE DER ABHAENGIGEN VARIABLEN Y

78.5
74.3
104.3
87.6
95.9
109.2
102.7
72.5
93.1
115.9
83.8
113.3
109.4

KREUZPRODUKTVEKTOR DER ABHAENGIGEN UND UNABHAENGIGEN VARIABLEN

1240.5
10032
62027.8
13981.5
34733.3

REGRESSIONSKOEFFIZIENTEN

62.4054
1.5511
0.510168
0.101909
-.144061

STATISTISCHE WERTE ZUR MULTIPLEN REGRESSION

VARIANZ-COVARIANZ MATRIX

4909.94	-50.5069	-50.6025	-51.66	-49.5971
-50.5069	0.554682	0.512657	0.554245	0.50529
-50.6025	0.512657	0.523869	0.525702	0.512129
-51.66	0.554245	0.525702	0.569586	0.516879
-49.5971	0.50529	0.512129	0.516879	0.502755

SUMME Y = 1240.5
QUADRATSUMME = 121088
VARIANZ VON Y = 2715.76

SUMME DER ABWEICHUNGS-QUADRATE	MITTLERE SUMME ABWEICHUNGS-QUADRATE	FREIHEITSGRADE	URSACHE
2667.9	666.975	4	INNERHALB DER REGRESSION
47.8636	5.98295	8	UM DIE REGRESSIONSSTEIGUNG

F-WERT = 111.479
STANDARDFEHLER DER SCHAETZUNG = 2.44601
MULTIPLE BESTIMMTHEITSMASSE = 0.982376 ODER = 0.973563
MULTIPLE KORRELATIONSKOEFFIZIENTEN = 0.991149 ODER = 0.986693

T-TEST WERTE

0.890603
2.08266
0.704858
0.135031
-.203174

```
STANDARDFEHLER DER REGRESSIONSKOEFFIZIENTEN
-------------------------------------------
      70.0709
      0.74477
      0.723788
      0.754709
      0.709052

BEOBACHTUNGSWERTE          GESCHAETZTE Y-WERTE          RESIDUEN
-----------------          -------------------          --------
      78.5                       78.4952                0.00476041
      74.3                       72.7888                  1.5112
     104.3                      105.971                  -1.67094
      87.6                       89.3271                 -1.7271
      95.9                       95.6492                 0.250756
     109.2                      105.275                   3.92544
     102.7                      104.149                  -1.44867
      72.5                       75.675                  -3.17499
      93.1                       91.7217                  1.37835
     115.9                      115.618                  0.281548
      83.8                       81.809                   1.99098
     113.3                      112.327                  0.972989
     109.4                      111.694                  -2.29433

DURBIN-WATSON TEST = 2.0526

VON NEUMANN VERHAELTNIS = 2.22365
```

52 Betriebswirtschaftslehre

521 Input-Output Analyse

Die betriebswirtschaftliche Anwendung des Input-Output Modells erstreckt sich vor allem auf die Gebiete der Kostenrechnung, und dort speziell auf die

- Kostenplanung und Kontrolle[11]
- Kostenumlage[12] und
- Budgetierung[13].

Weitere Anwendungen finden sich in den betriebswirtschaftlichen Verflechtungsmodellen der

- Produktionsplanung
- Materialplanung und des
- Soll-Ist-Vergleichs[14]

11 John L. Livingstone. Input-Output Analysis for Cost Accounting, Planning and Control. In: J. L. Livingstone (ed.), Management Planning and Control: Mathematical Models, New York 1970, p. 75–98.

12 Mattessich, Die wissenschaftlichen Grundlagen des Rechnungswesens, p. 227–229.

13 Farag, Input-Output Analysis, p. 92.

14 Günter Dietrich und Henry Stahl. Grundzüge der Matrizenrechnung. 5. Aufl., Leipzig 1968, S. 196–219.

Den betriebswirtschaftlichen Input-Output Modellen liegen für die technischen Koeffizienten die Annahmen der
- Linearität
- Homogenität und
- Nichtsubstitutionsfähigkeit

zugrunde[15].

Das folgende Beispiel zeigt die Budgetplanung einer Unternehmung mit drei Produktionszweigen[16]. Ausgangspunkt bildet das Input-Output Tableau zum Zeitpunkt T. Gefragt ist nach den Inputwerten zum Zeitpunkt $T + \Delta T$ der Produktionszweige und des primären Inputs, der hier durch die Rohmaterialkosten, Lohnkosten, Gemeinkosten und einen Gewinnanteil dargestellt wird. In der Vorausplanung werden die Planwerte für die Endbeträge des Verkaufs und des Lagerbestandes festgelegt. Die Berechnungen erfolgen mit dem in Abs. 512 zeigten BASIC-Programm. Liste der Eingabedaten:

```
LIST OF DATA AFTER END STATEMENT

"-------- BETRIEBSWIRTSCHAFTLICHES BEISPIEL -------"
3,2,4
0,30,60
10,0,90
25,25,0
60,50
70,80
100,150
70,50,30
50,55,60
20,40,15
25,50,45
"PROD.DEP-1","PROD.DEP-2","PROD.DEP-3"
"   SALES","INVENTORY"
"ROW.MATER.","LABOR COST","OVERHEAD","PROFIT"
200,100
170,30
300,100
```

15 Mattessich, Die wissenschaftlichen Grundlagen des Rechnungswesens, S. 225–227.

16 Die Beispieldaten sind entnommen aus: Farag, Input-Output Analysis, p. 92–97.

Programmergebnisse:

STARTTABLEAU

------- BETRIEBSWIRTSCHAFTLICHES BEISPIEL -------

----INPUT----	----OUTPUT----					
	PROD.DEP-1	PROD.DEP-2	PROD.DEP-3	SALES	INVENTORY	TOTAL
PROD.DEP-1	0	30	60	60	50	200
PROD.DEP-2	10	0	90	70	80	250
PROD.DEP-3	25	25	0	100	150	300
ROW.MATER.	70	50	30			150
LABOR COST	50	55	60			165
OVERHEAD	20	40	15			75
PROFIT	25	50	45			120
TOTAL	200	250	300	230	280	1260

ENDTABLEAU

------- BETRIEBSWIRTSCHAFTLICHES BEISPIEL -------

----INPUT----	----OUTPUT----					
	PROD.DEP-1	PROD.DEP-2	PROD.DEP-3	SALES	INVENTORY	TOTAL
PROD.DEP-1	0	44.3814	98.4681	200	100	442.849
PROD.DEP-2	22.1425	0	147.702	170	30	369.845
PROD.DEP-3	55.3562	36.9845	0	300	100	492.341
ROW.MATER.	154.997	73.9689	49.2341			278.2
LABOR COST	110.712	81.3658	98.4681			290.546
OVERHEAD	44.2849	59.1751	24.617			128.077
PROFIT	55.3562	73.9689	73.8511			203.176
TOTAL	442.849	369.845	492.341	670	230	2205.03

„Bei der Plankostenrechnung liegt der Schwerpunkt auf der Betriebskontrolle; wichtigstes Kontrollinstrument sind die Abweichungen der Ist- von den Sollkosten und den verrechneten Plankosten“[17].
Aus der Vielzahl der Systeme zur Abweichungsermittlung[18] soll hier ein Vierspaltungsverfahren gezeigt werden[19]. Die Gesamtabweichung wird in die
- Preisabweichung,
- Verbrauchsabweichung, aufgeteilt in
  - Mischungsabweichung
  - Ausbeuteabweichung und
- Beschäftigungsabweichung

unterteilt, wobei die kostentheoretische Relevanz hier nicht zur Diskussion steht.
Zur formelmässigen Darstellung der Abweichungsanalyse werden für die Kostenart i folgende Abkürzungen eingeführt[20]:

SM_i = Sollverbrauchsmenge je Einheit
SGM_i = SM_i x produzierte Einheiten = Sollgesamtverbrauchsmenge
PK_i = Plankosten je Einheit
MA_i = Mischungsanteil bei Standardmischung der Kostengruppe j
PKE_i = PK_i x SM_i = Plankosten einer produzierten Einheit
IGM_i = Istgesamtverbrauchsmenge
IGK_i = IGM_i x IK_i = Istgesamtkosten
IK_i = Istkosten je Einheit
GE_j = Gesamteinsatz der Kostengruppe j
ZGM_i = MA_i x GE_j = Zwischengesamtmengen = Istverbrauchsmengen bei Standardmischung
PA_i = Preisabweichung
MIA_i = Mischungsabweichung
AA_i = Ausbeuteabweichung
VA_i = Verbrauchsabweichung
BA_i = Beschäftigungsabweichung

17 Hanns-Martin Schönfeld. Kostenrechnung II. 7. Aufl., Stuttgart 1975, S. 38.
18 Karl Weber. Zur Abweichungsermittlung bei der Standardkostenrechnung. Industrielle Organisation 30 (1961) 9, S. 435–438.
19 Schönfeld, Kostenrechnung II, S. 39–40.
20 Schönfeld, Kostenrechnung II, S. 42.

Für die Kostenart i der Kostengruppe j innerhalb einer Kostenstelle bestimmen sich die Abweichungsarten nach den folgenden Formeln[21]:

$$\begin{aligned} PA_i &= IGM_i\,(IK_i - PK_i) \\ MIA_i &= PK_i\,(IGM_i - MA_i \cdot GE_j) \\ AA_i &= PK_i\,(SGM_i - MA_i \cdot GE_j) \\ VA_i &= PK_i\,(IGM_i - SGM_i) \\ BA_i &= (IGM_i \cdot IK_i) - (SGM_i \cdot PK_i) \end{aligned}$$

Der Mischungsanteil bei Standardmischung MA_i ergibt sich aus der Sollverbrauchsmenge einer Kostengruppe SMG_j dividiert durch die Sollverbrauchsmenge der Kostenart SM_i

$$MA_i = \frac{SMG_j}{SM_i} \text{ mit } SMG_j = \sum_{i=1}^{n_j} SM_i,$$

wobei n_j die Anzahl der Kostenarten der Kostengruppe j angibt.
Bei der Programmierung hat sich gezeigt, dass die in der Literatur übliche Darstellung der Abweichungsberechnung mit Matrizenoperationen[22] mit BASIC MAT-Statements nur teilweise zu verwirklichen ist. Die Zusammenfassung der Sollgesamtverbrauchsmengen SGM_i, Zwischengesamtmengen ZGM_i und Istgesamtverbrauchsmengen IGM_i in einer Mengenmatrix macht eine Abweichungsanalyse aus der Gesamtplankostenmatrix mit den verfügbaren MAT-Statements unmöglich. Wird auf diese Zusammenfassung verzichtet und werden die Zeilen der Mengenmatrix in Zeilenvektoren aufgeteilt, so bleibt der Rechenvorgang erhalten. Die Plankostenmatrix spaltet sich dadurch ebenfalls in Zeilenvektoren auf. Die umständlich zu programmierenden Zeilensubstraktionen innerhalb der Plankostenmatrix entfallen und die Abweichungsanalyse reduziert sich auf die Subtraktion von Vektoren, was leicht mit MAT-Statements dargestellt werden kann. Werden die Plan- und Istkosten der Kostenarten in die Hauptdiagonale von zwei N × N-Matrizen geschrieben, so ergeben sich durch Matrizenmultiplikation die

- Istgesamtkosten

$$IGK = IGM \cdot IK,$$

- Gesamtkosten auf Sollmengenbasis

$$SPK = SGM \cdot PK,$$

21 Schönfeld, Kostenrechnung II, S. 42–43; vgl. auch: Werner Frank und Rene Manes. A Standard Cost Application of Matrix Algebra. In: J. L. Livingstone (ed.), Management Planning and Control, p. 193–196.

22 Schönfeld, Kostenrechnung II, S. 43–45; Frank and Manes, A Standard Cost Application of Matrix Algebra, p. 197–202.

– Plankosten auf Istmengenbasis bei Standardmischung

$$ZPK = ZGM \cdot PK$$

und die

– Plankosten auf Istmengenbasis bei Istmischung

$$IPK = IGM \cdot PK.$$

Die einzelnen Abweichungsarten errechnen sich aus einfachen Vektorsubtraktionen.

$$PA = IGK - IPK$$
$$MIA = IPK - ZPK$$
$$AA = ZPK - SPK$$
$$VA = IPK - SPK = MIA + AA$$
$$BA = IGK - SPK$$

PA, MIA, AA, VA, BA, IGK, SPK, ZPK und IPK sind Zeilenvektoren mit den Feldgrenzen (N + O), wobei N die Zahl der variablen Kostenarten und O die Zahl der Gemeinkosten angeben.

Die Zusammenfassung von m_j Kostenarten zu j Kostengruppen wird in dem folgenden BASIC-Programm durch den Vektor E mit der Feldgrenze (M) gesteuert, wobei M die Anzahl der Kostengruppen angibt. Die Elemente des Vektors E enthalten die Zahl der Kostenarten je Kostengruppe.

Die Bedeutung der Programmvariablen ist durch Kommentare erläutert. Die Dateneingabe erfolgt nach Kostenarten, wobei in der Reihenfolge eingelesen wird

– Sollverbrauchsmenge
– Plankosten
– Istgesamtverbrauchsmenge und
– Istkosten.

Zusätzlich wird eine Texteingabe von maximal 13 Zeichen zur Kostenartbezeichnung verlangt. Die Textangabe zur Kostenstellenbezeichnung umfasst maximal 80 Zeichen. Die Kostenarten- und Kostenstellenbezeichnungen werden nicht mehr speziell in einem sequentiell organisierten File gespeichert.

```
1000 REM KOSTENABWEICHUNGSANALYSE
1005 WIDTH 8
1010 ! *** BEDEUTUNG DER VARIABLEN ***
1015 ! U = SOLLVERBRAUCHSMENGEN BEI STANDARDMISCHUNG        DIMENSION U(N+O)
1020 ! V = ISTVERBRAUCHSMENGEN BEI STANDARDMISCHUNG        DIMENSION V(N+O)
1025 ! C = ISTVERBRAUCHSMENGEN BEI ISTMISCHUNG             DIMENSION C(N+O)
1030 ! I = ISTKOSTENMATRIX                                 DIMENSION I(N+O,N+O)
1035 ! S = SOLLKOSTENMATRIX                                DIMENSION S(N+O,N+O)
1040 ! A = GESAMTISTKOSTENVEKTOR                           DIMENSION A(N+O)
1045 ! K = EINGABEMATRIX                                   DIMENSION K(N+O,2)
1050 !    K(N+O,1) = MISCHUNGSANTEIL BEI SOLLVERBRAUCH
1055 !    K(N+O,2) = PLANKOSTEN PRO EINHEIT
1060 ! L = SOLLVERBRAUCHSMENGE JE KOSTENGRUPPE           DIMENSION L(N1+O)
1065 ! G = GESAMTEINSATZ JE KOSTENGRUPPE                 DIMENSION G(N1+O)
1070 ! E = ANZAHL DER KOSTENARTEN JE KOSTENGRUPPE        DIMENSION E(N1+O)
1075 ! N = ANZAHL DER KOSTENARTEN OHNE GEMEINKOSTEN
1080 ! O = ANZAHL DER GEMEINKOSTENARTEN
```

```
100085 ! N1 = ANZAHL DER KOSTENGRUPPEN
100090 DIM K(7,2),A(7),I(7,7),S(7,7),L(7),G(7),V(7),U(7),W(7),E(7)
100095 DIM B(7),C(7),Z(7),Y$(8)*15,D(6)
100100 REM X9 = OBERE FELDGRENZE; HIER 7
100105 LET X9=7
100110 DIM A$*80,B$(7)*13
100115 REM EINLESEN DER KOSTENSTELLENBEZEICHNUNG
100120 READ A$
100125 IF A$="STOP" GOTO  2115
100130 REM EINLESEN DER ANZAHL DER KOSTENGRUPPEN UND DER GEMEINKOSTEN
100135 READ N1,O
100140 REM EINLESEN DER IN DER PERIODE PRODUZIERTEN EINHEITEN
100145 READ X
100150 REM EINLESEN DER ANZAHL DER KOSTENARTEN JE KOSTENGRUPPE
100155 MAT E=ZER(N1+O)
100160 MAT READ E
100165 REM BERECHNEN DER ANZAHL ALLER KOSTENARTEN
100170 MAT I=CON(N1+O,1)
100175 MAT N=E*I
100180 LET N=N-O
100185 IF N+O<=X9 GOTO  1200
100190 PRINT "DIMENSION ERHOEHEN, DA FELDER ZU KLEIN FUER EINGABEDATENMENGE"
100195 STOP
100200 REM REDIMENSIONIEREN ALLER FELDER AUF DIE AKTUELLEN GRENZEN
100205 MAT I=ZER(N+O,N+O)
100210 MAT S=ZER(N+O,N+O)
100215 MAT U=ZER(N+O)
100220 MAT V=ZER(N+O)
100225 MAT C=ZER(N+O)
100230 MAT I=ZER(N+O,N+O)
100235 MAT S=ZER(N+O,N+O)
100240 MAT A=ZER(N+O)
100245 MAT K=ZER(N+O,2)
100250 MAT B=ZER(N+O)
100255 MAT W=ZER(N+O)
100260 MAT Z=ZER(N+O)
100265 MAT L=ZER(N1+O)
100270 MAT G=ZER(N1+O)
100275 REM EINLESEN DER EINGABEDATEN
100280 ! REIHENFOLGE:
100285 ! SOLLVERBRAUCHSMENGE JE EINHEIT
100290 ! PLANKOSTEN JE EINHEIT
100295 ! ISTVERBRAUCHSMENGEN
100300 ! ISTKOSTEN
100305 ! BEZEICHNUNG DER KOSTENART MAXIMAL 10 ZEICHEN
100310 LET J1=1
100315 FOR J=1 TO N+O
100320  READ U(J),S(J,J),C(J),I(J,J),B$(J)
100325  LET K(J,2)=U(J)*S(J,J)
100330  IF J>E(J1) GOTO  1350
100335  LET L(J1)=L(J1)+U(J)
100340  LET G(J1)=G(J1)+C(J)
100345  GOTO  1365
100350  LET J1=J1+1
100355  LET E(J1)=E(J1-1)+E(J1)
100360  GOTO  1335
100365  LET K1=K1+K(J,2)
100370  LET U(J)=U(J)*X
100375 NEXT J
100380 REM BERECHNUNG DER MISCHUNGSANTEILE BEI SOLLVERBRAUCH
100385 LET J1=1
100390 FOR J=1 TO N+O
100395  IF J>E(J1) GOTO  1410
100400  LET K(J,1)=U(J)/(L(J1)*X)
100405  GOTO  1420
100410  LET J1=J1+1
100415  GOTO  1400
100420 NEXT J
100425 REM BERECHNEN ISTVERBRAUCHSMENGEN BEI STANDARDMISCHUNG
100430 LET J1=1
100435 FOR J=1 TO N+O
100440  IF J>E(J1) GOTO  1455
100445  LET V(J)=K(J,1)*G(J1)
100450  GOTO  1465
100455  LET J1=J1+1
100460  GOTO  1445
```

```
1465 NEXT J
1470 MAT Y$=("-------------- ")
1475 REM BERECHNUNG DER GESAMTISTKOSTEN JE KOSTENART
1480 MAT A=C*I
1485 REM BERECHNUNG DER PLANGESAMTKOSTEN AUF SOLLMENGENBASIS JE KOSTENART
1490 MAT B=U*S
1495 REM BERECHNUNG DER PLANKOSTEN AUF ISTMENGENBASIS BEI PLANMISCHUNG
1500 MAT Z=V*S
1505 REM BERECHNUNG DER PLANKOSTEN AUF ISTMENGENBASIS BEI ISTMISCHUNG
1510 MAT W=C*S
1515 PRINT "***KOSTENSTELLE***",A$
1520 PRINT
1525 REM DRUCKEN DER EINGABEDATEN
1530 PRINT " ","SOLLVERBRAUCH UND PLANKOSTEN"," "," ",
1535 PRINT "ISTVERBRAUCH UND ISTKOSTEN"
1540PRINT" ","JE EINHEIT  "," "," "," ","BEI PRODUKTION VON *";X;"* EINHEITEN"
1545 PRINT
1550PRINT"KOSTENART      ","SOLLVERBRAUCHS-","     PLANKOSTEN","     MISCHUNGS-",
1555 PRINT "     PLANKOSTEN","     ISTGESAMT-","     GESAMTIST-","     ISTKOSTEN"
1560PRINT" ","    MENGEN"," ","     ANTEIL","     PRO EINHEIT","     VERBRAUCHS-",
1565 PRINT "     KOSTEN"
1570 PRINT " "," "," "," "," ","      MENGEN"
1575 PRINT
1580 LET J1=1
1585 FOR J=1 TO N+O+1
1590  IF J>E(J1) GOTO  1635
1595  PRINT B$(J),U(J)/X,S(J,J),K(J,1),K(J,2),C(J),A(J),I(J,J)
1600  LET D(1)=D(1)+U(J)/X
1605  LET D(2)=D(2)+S(J,J)
1610  LET D(3)=D(3)+K(J,2)
1615  LET D(4)=D(4)+C(J)
1620  LET D(5)=D(5)+A(J)
1625  LET D(6)=D(6)+I(J,J)
1630  GOTO  1710
1635  LET K2=K2+D(4)
1640  LET K3=K3+D(5)
1645  LET K4=K4+D(1)
1650  LET K5=K5+D(2)
1655  LET K6=K6+D(6)
1660  IF E(J1)-E(J1-1)=1 GOTO  1685
1665  MAT PRINT Y$;
1670  PRINT "GRUPPENSUMMEN",D(1),D(2),1,D(3),D(4),D(5),D(6)
1675  PRINT "PLANVERBRAUCH DER KOSTENGRUPPE", TAB (50);L(J1)*X
1680  PRINT "GESAMTEINSATZ DER KOSTENGRUPPE", TAB (50);G(J1)
1685  LET J1=J1+1
1690  PRINT
1695  MAT D=ZER
1700  IF J1>N1+O GOTO  1715
1705  GOTO  1595
1710 NEXT J
1715 PRINT "GESAMT",K4,K5," ",K1,K2,K3,K6
1720 LET K1=0
1725 LET K2=0
1730 LET K3=0
1735 LET K4=0
1740 LET K5=0
1745 LET K6=0
1750 REM BERECHNUNG DER EINZELABWEICHUNGEN
1755 ! PREISABWEICHUNG IN C
1760 MAT C=A-W
1765 ! MISCHUNGSABWEICHUNG IN W
1770 MAT W=W-Z
1775 ! AUSBEUTEABWEICHUNG IN V
1780 MAT V=Z-B
1785 ! BESCHAEFTIGUNGSABWEICHUNG IN U
1790 MAT U=A-B
1795 ! VERBRAUCHSABWEICHUNG
1800 MAT A=W+V
1805 REM AUSGABE DER KOSTENABWEICHUNGEN
1810 PAGE
1815 PRINT "***KOSTENSTELLE***",A$
1820 PRINT
1825 PRINT "KOSTENART","PREIS-","MISCHUNGS-","AUSBEUTE-","VERBRAUCHS-",
1830 PRINT "GESAMT-"
1835 PRINT " ",
1840 FOR J=1 TO 5
```

```
1845  PRINT "ABWEICHUNG",
1850 NEXT J
1855 PRINT
1860 LET J1=1
1865 PRINT
1870 FOR J=1 TO N+O+1
1875  IF J>E(J1) GOTO  1915
1880  PRINT B$(J),C(J),W(J),V(J),A(J),U(J)
1885  LET D(1)=D(1)+C(J)
1890  LET D(2)=D(2)+W(J)
1895  LET D(3)=D(3)+V(J)
1900  LET D(4)=D(4)+A(J)
1905  LET D(5)=D(5)+U(J)
1910  GOTO  1955
1915  IF E(J1)-E(J1-1)=1 GOTO  1930
1920  MAT PRINT Y$(6);
1925  PRINT "GESAMT",D(1),D(2),D(3),D(4),D(5)
1930  MAT D=ZER
1935  LET J1=J1+1
1940  PRINT
1945  IF J1>N1+O GOTO  1960
1950  GOTO  1880
1955 NEXT J
1960 IF O=0 GOTO  2105
1965 PRINT
1970 PRINT
1975PRINT"BEI VARIABLEN GEMEINKOSTEN IST DIE AUFTEILUNG DER VERBRAUCHSABWEIC";
1980 PRINT "HUNG IN"
1985 PRINT "MISCHUNGS- UND AUSBEUTEABWEICHUNG NICHT ZU BERUECKSICHTIGEN"
1990 PRINT
1995PRINT"BEI FIXKOSTEN IST NUR DIE GESAMTABWEICHUNG (BESCHAEFTIGUNGSABWEICH";
2000 PRINT "UNG) ZU BERUECKSICHTIGEN"
2005 DATA "MANAGEMENT PLANNING AND CONTROL"
2010 DATA 2, 0
2015 DATA 10000
2020 DATA  2, 3
2025 DATA 3, 2, 32000, 2.1, "MATERIAL-1"
2030 DATA 2, 4, 19500, 4.05, "MATERIAL-2"
2035 DATA 0.25, 3, 2480, 3, "LOHN-1"
2040 DATA 0.5, 2.5, 5100, 2.5, "LOHN-2"
2045 DATA 2, 2, 20500, 2.05, "LOHN-3"
2050 DATA "BEISPIEL AUS SCHOENFELD BAND II"
2055 DATA 2,2
2060 DATA 1000
2065 DATA 2,2,1,1
2070 DATA 3,2,3200,2.10,"MATERIAL A"
2075 DATA 2,4,1950,4.05,"MATERIAL B"
2080 DATA 0.5,2.5,510,2.5,"LOHN    I"
2085 DATA 2,2,2050,2.05,"LOHN    II"
2090 DATA 2.5,4,2560,4.2,"VAR.GEMKO."
2095 DATA 2.5,1,2560,1.25,"FIX.GEMKO."
2100 DATA "STOP"
2105 PAGE
2110 GOTO  1120
2115 STOP
2120 END
```

KOSTENSTELLE MANAGEMENT PLANNING AND CONTROL

KOSTENART	SOLLVERBRAUCH UND PLANKOSTEN JE EINHEIT: SOLLVERBRAUCHS-MENGEN	PLANKOSTEN	MISCHUNGS-ANTEIL	PLANKOSTEN PRO EINHEIT	ISTVERBRAUCH UND ISTKOSTEN BEI PRODUKTION VON * 10000 * EINHEITEN: ISTGESAMT-VERBRAUCHS-MENGEN	GESAMTIST-KOSTEN	ISTKOSTEN
MATERIAL-1	3	2	0.6	6	32000	67200	2.1
MATERIAL-2	2	4	0.4	8	19500	78975	4.05
GRUPPENSUMMEN	5	6	1	14	51500	146175	6.15
PLANVERBRAUCH DER KOSTENGRUPPE			50000				
GESAMTEINSATZ DER KOSTENGRUPPE			51500				
LOHN-1	0.25	3	0.0909091	0.75	2480	7440	3
LOHN-2	0.5	2.5	0.181818	1.25	5100	12750	2.5
LOHN-3	2	2	0.727273	4	20500	42025	2.05
GRUPPENSUMMEN	2.75	7.5	1	6	28080	62215	7.55
PLANVERBRAUCH DER KOSTENGRUPPE			27500				
GESAMTEINSATZ DER KOSTENGRUPPE			28080				
GESAMT	7.75	13.5		20	79580	208390	13.7

KOSTENSTELLE MANAGEMENT PLANNING AND CONTROL

KOSTENART	PREIS-ABWEICHUNG	MISCHUNGS-ABWEICHUNG	AUSBEUTE-ABWEICHUNG	VERBRAUCHS-ABWEICHUNG	GESAMT-ABWEICHUNG
MATERIAL-1	3200	2200	1800	4000	7200
MATERIAL-2	975	-4400	2400	-2000	-1025
GESAMT	4175	-2200	4200	2000	6175
LOHN-1	0	-218.182	158.182	-60	-60
LOHN-2	0	-13.6364	263.636	250	250
LOHN-3	1025	156.364	843.636	1000	2025
GESAMT	1025	-75.4545	1265.45	1190	2215

SOLLVERBRAUCH UND PLANKOSTEN JE EINHEIT

ISTVERBRAUCH UND ISTKOSTEN BEI PRODUKTION VON * 1000 * EINHEITEN

KOSTENART	SOLLVERBRAUCHS-MENGEN	PLANKOSTEN	MISCHUNGS-ANTEIL	PLANKOSTEN PRO EINHEIT	ISTGESAMT-VERBRAUCHS-MENGEN	GESAMTIST-KOSTEN	ISTKOSTEN
MATERIAL A	3	2	0.6	6	3200	6720	2.1
MATERIAL B	2	4	0.4	8	1950	7897.5	4.05
GRUPPENSUMMEN	5	6	1	14	5150	14617.5	6.15
PLANVERBRAUCH DER KOSTENGRUPPE			5000				
GESAMTEINSATZ DER KOSTENGRUPPE			5150				
LOHN I	0.5	2.5	0.2	1.25	510	1275	2.5
LOHN II	2	2	0.8	4	2050	4202.5	2.05
GRUPPENSUMMEN	2.5	4.5	1	5.25	2560	5477.5	4.55
PLANVERBRAUCH DER KOSTENGRUPPE			2500				
GESAMTEINSATZ DER KOSTENGRUPPE			2560				
VAR.GEMKO.	2.5	4	1	10	2560	10752	4.2
FIX.GEMKO.	2.5	1	1	2.5	2560	3200	1.25
GESAMT	12.5	15.5		31.75	12830	34047	16.15

KOSTENSTELLE BEISPIEL AUS SCHOENFELD BAND II

KOSTENART	PREIS-ABWEICHUNG	MISCHUNGS-ABWEICHUNG	AUSBEUTE-ABWEICHUNG	VERBRAUCHS-ABWEICHUNG	GESAMT-ABWEICHUNG
MATERIAL A	320	220	180	400	720
MATERIAL B	97.5	-440	240	-200	-102.5
GESAMT	417.5	-220	420	200	617.5
LOHN I	0	-5	30	25	25
LOHN II	102.5	4	96	100	202.5
GESAMT	102.5	-1	126	125	227.5
VAR.GEMKO.	512	0	240	240	752
FIX.GEMKO.	640	0	60	60	700

BEI VARIABLEN GEMEINKOSTEN IST DIE AUFTEILUNG DER VERBRAUCHSABWEICHUNG IN MISCHUNGS- UND AUSBEUTEABWEICHUNG NICHT ZU BERUECKSICHTIGEN

BEI FIXKOSTEN IST NUR DIE GESAMTABWEICHUNG (BESCHAEFTIGUNGSABWEICHUNG) ZU BERUECKSICHTIGEN

LP-Modelle sind generell durch eine lineare Zielfunktion, lineare Restriktionen und Nichtnegativitätsbedingungen charakterisiert.[23] Sie lassen sich allgemein in der Form

$$(1)\quad Z = \sum_{j=1}^{N} c_j x_j \stackrel{!}{=} \{\max \mid \min\}$$

$$(2)\quad \sum_{j=1}^{N} a_{ij} x_j \; \varrho_i \; b_i,\; i = 1(1)M,\; \varrho_i \in \{\leq \mid = \mid \geq\}$$

$$(3)\quad x_j \geq 0,\; j = 1(1)N$$

darstellen, wobei

x_j Entscheidungs-(Struktur-)variable,
c_j Koeffizienten der Zielfunktion,
a_{ij} Koeffizienten der Nebenbedingungen und
b_i Beschränkungswerte

bezeichnen. Die Zahl der Entscheidungsvariablen beträgt N, diejenige der Beschränkungswerte M.

Lineare Optimierungsaufgaben lassen sich mit Hilfe der Simplexmethode[24] lösen, wobei – ausgehend von einer ersten zulässigen Basislösung – sukzessive Lösungsverbesserungen erarbeitet und schliesslich (falls existent) die Optimallösung gewonnen wird.

Bei Aufgaben, die ausschliesslich Nebenbedingungen vom LE-Typ ($\leq$) umfassen und deren Beschränkungswerte b_i alle nichtnegativ sind, kann als erste zulässige Basislösung unmittelbar die Nullösung angegeben werden. Bei Aufgaben mit gemischten Nebenbedingungen, Restriktionen vom EQ- und/oder GE-Typ ($=, \geq$) umfassend, kann eine erste zulässige Basislösung über eine Hilfsaufgabe bestimmt werden (Zweiphasenmethode).[25]

23 Heiner Müller-Merbach. Programmierung, lineare. In: Erwin Grochla und Waldemar Wittmann (Hrsg.). Handwörterbuch der Betriebswirtschaft. 4. Aufl., Bd. 2. Stuttgart 1975, Sp. 3217–3227.

24 George B. Dantzig. Lineare Programmierung und Erweiterungen. Berlin 1966, S. 110–170.

25 Dantzig, Lineare Programmierung und Erweiterungen, S. 118–129; G. Hadley. Linear Programming. Reading 1962, p. 149–158, 207–215; Werner Dück und Manfred Bliefernich (Hrsg.). Operationsforschung 2: Mathematische Grundlagen, Methoden und Modelle. Berlin 1971, S. 152–156; Peter Haupt und Dieter Lohse. Grundlagen und Anwendungen der Linearen Optimierung. Essen 1975, S. 197–21; Heinz Körth, Carl Otto, Walter Runge und Manfred Schoch. Lehrbuch der Mathematik für Wirtschaftswissenschaften. Opladen 1972, S. 295–308.

Bei der Zweiphasenmethode besteht die erste Arbeitsstufe in der Problemstrukturierung. Zu bestimmen sind

- Zielfunktion mit
  N Entscheidungsvariablen,
- Restriktionensystem mit insgesamt
  M Nebenbedingungen, davon
  L Nebenbedingungen vom LE-Typ,
  E Nebenbedingungen vom EQ-Typ und
  G Nebenbedingungen vom GE-Typ.

Die Nebenbedingungen sind im Restriktionensystem in der vorerwähnten Reihenfolge blockweise anzuordnen. Weiterhin ist die LP-Aufgabe in die Normalform überzuführen.
Für Maximumaufgaben stellt sich die Normalform wie folgt dar:

$$\max \{c^T x \mid Ax = b,\ x \geq 0,\ b \geq 0\}$$

Treten Beschränkungswerte $b_i < 0$ auf, so kann die Normalform durch Multiplikation mit -1 erreicht werden.
Nebenbedingungen, die keine (EQ-Typ) oder negative (GE-Typ) Schlupfvariable enthalten, sind durch künstliche Variable zu ergänzen. Insgesamt sind

$$H = E + G$$

künstliche Variable einzuführen.

Die zweite Arbeitsstufe (Phase I des Lösungsprozesses) besteht in der Erklärung einer – auf dem Restriktionensystem der Ausgangsaufgabe basierenden – Hilfsaufgabe mit H künstlichen Variablen und der Zielfunktion

$$Z^* = \sum_{h=1}^{H} (-1)\, x_h \stackrel{!}{=} \max$$

Zur Bestimmung von max Z^* kann der Simplexalgorithmus verwendet werden, wobei sich – nach Umsetzung der Zielfunktion in die kanonische Form[26] – das aus Abbildung 30 ersichtliche Ausgangstableau ergibt. Hierbei sind S_j und S_B nach den Formeln

$$S_j = \sum_{i=L+1}^{M} (-1)\, a_{ij} \qquad j = 1(1)N + G$$

26 Vgl. Haupt und Lohse, Grundlagen und Anwendungen der Linearen Optimierung, S. 202.

Zeile		Basis-variable X (J)	N Problem-variable X (J)	G Negative Schlupfvariable X (J)	L Positive Schlupfvariable X (J)	H Künstliche Schlupfvariable X (J)	Beschränkungswerte b (I)
I		J \ J	1 ... N	(N + 1) ... (N + G)	(N + G + 1) ... (N + G + L)	(N + G + L + 1) ... (N + G + E + L)	B
1	LE	N + G + 1	$a_{11} \cdots a_{1N}$		1 ⋯ 0		b_1
.		.	.		. ⋱ .		.
L		N + G + L	$a_{L1} \cdots a_{LN}$		0 ⋯ 1		b_L
L + 1	EQ	N + G + L + 1	$a_{L+1,1} \cdots$			1 ⋯ 0	b_{L+1}
.		.	.			. ⋱ .	.
L + E		N + G + L + E	$a_{L+E,1} \cdots$			0 ⋯ 1 ⋯ 0	b_{L+E}
L + E + 1	GE	N + G + L + E + 1	$a_{L+E+1,1} \cdots$	−1 ⋯ 0		0 ⋯ 1 ⋯ 0	b_{L+E+1}
.		.	.	. ⋱ .		. ⋱ .	.
M = L + E + G		N + H + L	$a_{M1} \cdots a_{MN}$	0 ⋯ −1		0 ⋯ 1	b_M
M + 1		Ausgangsaufgabe	$c_1 \cdots c_N$				0
M + 2		Hilfsaufgabe	$S_1 \cdots S_N$	1 ⋯ 1			S_B

Die freien Felder werden mit 0 belegt.

Abbildung 30. Ausgangstableau zu Phase I, künstliche Variable in Basis

und

$$S_B = \sum_{i=L+1}^{M} (-1) b_i$$

zu s berechnen.

Ergibt sich für die Hilfsaufgabe als Optimallösung ein Zielfunktionswert von $Z^* < 0$, so besitzt die Ausgangsaufgabe keine zulässige Lösung. Bei max $Z^* = 0$ liegt eine zulässige Ausgangslösung vor.[27]

Vermerkt sei, dass die Bearbeitung der Hilfsaufgabe im allgemeinen mindestens H Simplexschritte erfordert, da bei jedem Simplexschritt eine Basisvariable ausgetauscht wird und der Anfangsvektor der Hilfsaufgabe die H künstlichen Variablen enthält, die bei einer nicht ausgearteten optimalen Lösung mit $Z^* = 0$ nicht auftreten dürfen.

In einer dritten Arbeitsstufe ist – ausgehend von der ermittelten zulässigen Ausgangslösung und unter Verwendung des Simplexalgorithmus – die Optimallösung für die Ausgangsaufgabe zu bestimmen (Phase II des Lösungsprozesses). Die hierzu erforderliche Zahl von Simplexschritten ist im allgemeinen gering, da die aufgrund der Hilfsaufgabe erhaltene zulässige Basislösung meist schon eine gute Annäherung des Optimums und bisweilen sogar bereits die optimale Lösung ist.

In dem wiedergegebenen BASIC-Programm[28] werden die drei Arbeitsstufen

- Problemstrukturierung
- Lösung der Hilfsaufgabe (Phase I)
- Lösung der Ausgangsaufgabe (Phase II)

deutlich ausgewiesen. Die Steuerung des Tabellenausdrucks (8 Druckspalten) erfolgt FOSBIC-spezifisch mit *8.

Reihenfolge der Dateneingabe:

- Steuerungswerte
  Z (1 für Maximumaufgabe, −1 für Minimumaufgabe)
  T (9 mit, −9 ohne Tabellenausdruck)
- Strukturparameter
  N, M, L, E und G, wobei M = L + G = E sein muss.
- Restriktionskoeffizienten, a_{ij}, für die blockweise angeordneten Nebenbedingungen vom
  LE-Typ (positive Schlupfvariable), i = 1(1)L
  EQ-Typ (keine Schlupfvariable) i = (L + 1)(1)(L + E)
  GE-Typ (negative Schlupfvariable i = (L + E + 1)(1)(L + E + G)
  Die Dateneingabe hat in der angegebenen Restriktionsfolge zu geschehen, wobei Werte von $a_{ij} = 0$ explizit anzugeben sind.

227 Dantzig, Lineare Programmierung und Erweiterungen, S. 123; Hadley, Linear Programming, p. 150–151.

228 Ursprungsversion: Dartmouth Public Files/Michigan State University.

- Beschränkungswerte, b_i, in der Folge i = 1(1)M
- Zielfunktionskoeffizienten, c_j, in der Folge J = 1(1)N
- Problembeschreibung, alphanumerische Zeichenfolge (minimal 66, maximal 70 Zeichen)

T = 9 bewirkt den Druck des für die Zweiphasenmethode typischen Ausgangstableaus; es enthält in der ersten Spalte die Nummern der Basisvariablen X(J) und in den weiteren Spalten die Entscheidungs-, Überschuss- und Schlupfvariable betreffenden Koeffizienten. Die letzte Spalte – programmintern wird der Tabellenausdruck auf insgesamt 8 Spalten beschränkt – enthält die Beschränkungswerte b_i. Weiterhin werden in der vorletzten bzw. letzten Zeile die für Phase II bzw. Phase I relevanten Zielfunktionskoeffizienten ausgegeben. Das entsprechend aufgebaute Schlusstableau enthält in der vorletzten Tabellenzeile die Schattenpreise. Das Programm lässt die sukzessive Abarbeitung einer beliebigen Zahl von LP-Aufgaben zu. Der normale Programmabschluss wird durch Eingabe von Z = 99, T = 9 im letzten DATA- resp. data-Statement gesteuert.

```
REM     LINEARE PROGRAMMIERUNG - ZWEI-PHASEN METHODE
!
!   URSPRUNGSVERSION DARTMOUTH PUBLIC FILES/MICHIGAN STATE UNIVERSITY,
!   MODIFIZIERT
!
!       NOMENKLATUR
!       Z  =  PROBLEMART, MAX=1, MIN=-1
!       T  =  TABELLENDRUCK, JA=9, NEIN=-9
!       N  =  ZAHL DER ENTSCHEIDUNGSVARIABLEN (STRUKTURVARIABLEN)
!       M  =  ZAHL DER NEBENBEDINGUNGEN
!       L  =  ZAHL DER NEBENBEDINGUNGEN MIT  ZAHL DER POSITIVEN
!             SCHLUPFVARIABLEN
!       E  =  ZAHL DER NEBENBEDINGUNGEN MIT =
!       G  =  ZAHL DER NEBENBEDINGUNGEN MIT Ö, ZAHL DER NEGATIVEN
!             SCHLUPFVARIABLEN (UEBERSCHUSSVARIABLEN)
!       H  =  E + G = ZAHL DER KUENSTLICHEN VARIABLEN
!       B  =  M + N + G + 1
!       W  =  M
!       (B+1)*(W+2) = SPALTEN + ZEILEN = ERFORDERLICHE GROESSE VON A
!       A  =  SIMPLEXTABLEAU
!
!       DATENEINGABE
!       RESTRIKTIONEN IN REIHENFOLGE DER TYPEN <=, =, >= IN DAS
!       AUSGANGSTABLEAU UEBERNEHMEN
!       A) PARAMETER Z BIS G  - PROBLEMSTRUKTUR
!       B) KOEFFIZIENTEN
!            1. RESTRIKTIONSKOEFFIZIENTEN, ZEILENWEISE IN DER FOLGE
!               DER RESTRIKTIONEN VOM TYP <=, =, >=. KEINE ANGABE
!               DER SCHLUPFVARIABLEN, KUENSTLICHEN VARIABLEN
!               ODER HILFSVARIABLEN.
!            2. BESCHRAENKUNGSWERTE IN DER REIHENFOLGE DER ZEILEN
!               VON 1.
!            3. ZIELFUNKTIONSKOEFFIZIENTEN
!       C) PROBLEMUMSCHREIBUNG, STANDARDISIERTE STRINGLAENGE 70 ZEICHEN,
!          STRINGKLAMMERN IN SP. 1 UND 70 ODER ENTSPRECHEND VERSCHOBEN
!
REM*****PROBLEMSTRUKTUR*************************************************
LET Q = 99999
LET A7=15
LET A8=16
REM A7 UND A8 GEBEN DIE MAXIMALE ZEILEN- UND SPALTENZAHL VON MAT A AN
!   R$ TEXTVARIABLE ZUR PROBLEMUMSCHREIBUNG, 70 ZEICHEN
DIM A(15,16), B$(3)*5, R$*70
LET B$(2)="MUMAU"
LET B$(3)="FGABE"
GOTO 350
10 READ N,M,L,E,G
PAGE
PRINT "PROBLEM NR."; P1
PRINT "***************"
PRINT
PRINT "PROBLEMART ";B$(1);B$(2);B$(3)
PRINT
PRINT "    ZAHL DER ENTSCHEIDUNGSVARIABLEN", TAB(41);N
PRINT "    ZAHL DER NEBENBEDINGUNGEN", TAB(41);M
PRINT "             DAVON LE-TYP", TAB(50);L
PRINT "             DAVON EQ-TYP", TAB(50);E
PRINT "             DAVON GE-TYP", TAB(50);G
PRINT
LET Z = -Z
IF L+E+G=M GOTO 20
PRINT "PROGRAMMABBRUCH - BEDINGUNG M = L+E+G NICHT EINGEHALTEN"
GOTO 350
```

```
20 LET B=N+M+G+1
   LET W=M
   MAT  A = ZER(W+2,B+1)
   IF (B+1)*(W+2) < A7*A8 GOTO 30
   PRINT "PROGRAMMABBRUCH - PROBLEM ZU GROSS"
30 IF B+1 > A8 GOTO 40
   IF W+2 < A7 GOTO 50
40 PRINT "PROGRAMMABBRUCH - DIM-STATEMENTAENDERUNG AUF A(";W+2;",";B+1;
   PRINT ") NOETIG"
   GOTO 350
   REM*****PHASE I*************************************************************
50 LET M=M-1                              ! INITIALISIERUNG
   LET H=1
   FOR I=0 TO M
      FOR J=1 TO N
         READ A(I+1,J+1)
      NEXT J
   NEXT I
   FOR I=0 TO M
      READ A(I+1,B+1)
   NEXT I
   FOR J=1 TO N
      READ A(W+1,J+1)
      LET A(W+1,J+1) = Z*A(W+1,J+1)
   NEXT J
   FOR K=1 TO M+1                         ! TABLEAUAUFBAU
      LET A(K,N+G+K+1)=1
      LET A(K,1)=K+N+G
   NEXT K
   IF G <> 0 GOTO 60
   IF E = 0 GOTO 90
   GOTO 70
60 FOR K=L+E+1 TO M+1
      LET A(K,K+N-L-E+1)=-1
   NEXT K
70 LET W=W+1
   LET Q=0
   FOR J=1 TO N+G
      LET S=0
      FOR I=M-G-E+1 TO M
         LET S=S+A(I+1,J+1)
      NEXT I
      LET A(W+1,J+1)=-S
      IF A(W+1,J+1) > Q GOTO 80
      LET Q=A(W+1,J+1)
      LET C=J
80 NEXT J
   LET S=0
   FOR J=M-G-E+1 TO M
      LET S=S+A(J+1,B+1)
   NEXT J
   LET A(W+1,B+1)=-S
90 READ R$
   PRINT R$
   PRINT
   PRINT "VARIABLENBEZEICHNUNG", TAB(31),"NUMMERN"
   PRINT "ENTSCHEIDUNGSVARIABLE"; TAB(31); H; TAB(34); "-"; TAB(39); N
   IF G=0 GOTO 110
   IF G > 1 GOTO 100
   PRINT "     NEGATIVE SCHLUPFVARIABLE", TAB(31); N+1
   GOTO 110
100 PRINT "     NEGATIVE SCHLUPFVARIABLE"; TAB(31); N+1;
   PRINT TAB(34); "-", TAB(39); N+G
110 IF L=0 GOTO 130
   IF L > 1 GOTO 120
   PRINT "     POSITIVE SCHLUPFVARIABLE", TAB(31); N+G+1
```

```
   GOTO 130
120 PRINT "     POSITIVE SCHLUPFVARIABLE"; TAB(31); N+G+1;
   PRINT TAB(34), "-", TAB(39); N+G+L
130 IF G+E=0 GOTO 150
   IF G+E > 1 GOTO 140
   PRINT "     KUENSTLICHE VARIABLE", TAB(31); N+G+L+1
   GOTO 150
140 PRINT "     KUENSTLICHE VARIABLE"; TAB(31); N+G+L+1;
   PRINT TAB(34), "-", TAB(39); B-1
150 IF T=-9 GOTO 160
   PRINT "              VARIABLE IN ENTSPRECHENDER REIHENFOLGE IN TABLEAUS,"
   PRINT "              AB SPALTE 2, OHNE LETZTE SPALTE (B-VEKT./PRIMALLOES.)"
   GOSUB 390
160 PRINT                              ! PIVOTIERUNG
   IF Q=99999 GOTO 240
170 IF Q=0 GOTO 260
   GOTO 300
180 LET H=H+1
   LET Q=10**37
   LET R=-1
   FOR I=0 TO M
      IF A(I+1,C+1) <= 0 GOTO 190
      IF A(I+1,B+1)/A(I+1,C+1) > Q GOTO 190
      LET Q = A(I+1,B+1)/A(I+1,C+1)
      LET R=I
190 NEXT I
   IF R >= -0.5 GOTO 210
   PRINT "LOESUNG UNBEGRENZT"
   IF T=-9 GOTO 200
   GOSUB 390
200 GOTO 350
210 LET P=A(R+1,C+1)
   LET A(R+1,1) = C
   FOR J=1 TO B
      LET A(R+1,J+1) = A(R+1,J+1)/P
   NEXT J
   FOR I =0 TO W
      IF I=R GOTO 230
      FOR J=1 TO B
          IF J=C GOTO 220
          LET A(I+1,J+1) = A(I+1,J+1)-A(R+1,J+1)*A(I+1,C+1)
          IF ABS(A(I+1,J+1)) > 0.00001 GOTO 220
          LET A(I+1,J+1)=0
220      NEXT J
230 NEXT I
   FOR I=0 TO W
      LET A(I+1,C+1)=0
   NEXT I
   LET A(R+1,C+1)=1
240 LET Q=0
   FOR J=1 TO N+G+L
      IF A(W+1,J+1) > Q GOTO 250
      LET Q=A(W+1,J+1)
      LET C=J
250 NEXT J
   GOTO 170
   REM*****PHASE II*************************************************************
260 IF W=H+1 GOTO 290                  ! LOESBARKEITSTEST
   LET W=W-1
   IF A(W+2,B+1) < 0.000001 GOTO 270
   PRINT "KEINE ZULAESSIGE LOESUNG"
   GOTO 350
270 FOR I=0 TO M
      IF A(I+1,1) <= N+G+L GOTO 280
      FOR J=1 TO B
          LET A(I+1,J+1)=0
```

```
        NEXT J
280 NEXT I
    GOTO 240
290 PRINT "LOESUNG - PRIMALPROBLEM"   ! PIVOTIERUNG/ERGEBNISAUSDRUCK
300 IF Q=0 GOTO 310
    IF T=-9 GOTO 330
    PRINT "BASIS VOR ITERATION"; H
310 PRINT "      VARIABLE X(J)", TAB(24); "WERT"
    FOR I=0 TO M
       IF A(I+1,1)=0 GOTO 320
       PRINT A(I+1,1),A(I+1,B+1)
320 NEXT I
330 IF Q <> 0 GOTO 180
    IF T=-9 GOTO 340
    PRINT "SCHATTENPREISE (LOESUNG - DUALPROBLEM)"
    PRINT "      VARIABLE X(J)", TAB(24); "WERT"
    FOR J=N+1 TO B-G-1
       PRINT J,A(W+1,J+1)
    NEXT J
340 PRINT
    PRINT "      ZIELFUNKTIONSWERT ="; -Z*A(W+1,B+1); "NACH"; H-1; "ITERAT."
    PRINT
    IF T=-9 GOTO 350
    GOSUB 390
350 READ Z,T
    LET P1=P1+1
360 IF Z*10+T=999 GOTO 999            ! PROGRAMMABSCHLUSS BEI Z=99 UND T=9
    IF Z=1 GOTO 370
    LET B$(1)=" MINI"
    GOTO 380
370 LET B$(1)=" MAXI"
380 IF ABS(Z)+ABS(T)=10 GOTO 10
    PRINT "DATENEINGABE KONTROLLIEREN"
    GOTO 999
390 REM TABLEAUDRUCK
    IF B > 7 GOTO 400
    PRINT
    PRINT "TABLEAU NACH"; H-1; "ITERATIONEN, BASISVARIABLE IN SPALTE 1"
    MAT PRINT A,
400 RETURN
!
! BEISPIEL EINER DATENEINGABE FUER EIN LP-PROBLEM
!
! PARAMETER  Z   T   N   M  DER PROBLEMSTRUKTUR
    DATA  -1,  9,  2,  3
!
! NEBENBEDINGUNGEN LE  EQ  >=
    DATA          1,  1,  1
!
! KOEFFIZIENTEN DER RESTRIKTIONEN
    DATA  0.6,  0.1      ! LE-TYP
    DATA  1.0,  1.0      ! EQ-TYP
    DATA  0.2,  0.8      ! GE-TYP
!
! BESCHRAENKUNGSVEKTOR
    DATA  0.3,  1.0,  0.4
!
! KOEFFIZIENTEN DER ZIELFUNKTION
    DATA  1.0,  5.0
!
! PROBLEMUMSCHREIBUNG
    DATA "AUFGABE AUS HAUPT/LOHSE, GRUNDLAGEN UND ANWENDUNG DER LINEAREN OPT. "
!
! ENDE DER DATENEINGABE EINES LP-PROBLEMS
!
 999 END LIST DATA

LIST OF DATA AFTER END STATEMENT


-1,-9,2,3
1,1,1
0.6,0.1,1,1,0.2,0.8,0.3,1,0.4,1,5
"BEISPIEL HAUPT/LOHSE OHNE TABLEAU- UND ZWISCHENSCHRITTAUSDRUCK       "
1,+9,4,4
2,2,0
3,2,1,4,2,1,5,1,2,6,-4,8,1,3,-2,4,6,4,0,0,1,1.5,5,2
"BEISPIEL MIT RESTRIKTIONEN VOM LE- UND EQ-TYP                        "
99,9
*
```

```
PROBLEM NR. 1
**************

PROBLEMART  MINIMUMAUFGABE

    ZAHL DER ENTSCHEIDUNGSVARIABLEN      2
    ZAHL DER NEBENBEDINGUNGEN            3
            DAVON LE-TYP                      1
            DAVON EQ-TYP                      1
            DAVON GE-TYP                      1

AUFGABE AUS HAUPT/LOHSE, GRUNDLAGEN UND ANWENDUNG DER LINEAREN OPT.

VARIABLENBEZEICHNUNG          NUMMERN
ENTSCHEIDUNGSVARIABLE         1  -    2
     NEGATIVE SCHLUPFVARIABLE 3
     POSITIVE SCHLUPFVARIABLE 4
     KUENSTLICHE VARIABLE     5  -         6
            VARIABLE IN ENTSPRECHENDER REIHENFOLGE IN TABLEAUS,
            AB SPALTE 2, OHNE LETZTE SPALTE (B-VEKT./PRIMALLOES.)

TABLEAU NACH 0 ITERATIONEN, BASISVARIABLE IN SPALTE 1
       4            0.6          0.1           0          1          0          0          0.3
       5            1            1             0          0          1          0          1
       6            0.2          0.8          -1          0          0          1          0.4
       0            1            5             0          0          0          0          0
       0           -1.2         -1.8           1          0          0          0         -1.4
```

```
BASIS VOR ITERATION 1
     VARIABLE X(J)      WERT
        4               0.3
        5               1
        6               0.4
BASIS VOR ITERATION 2
     VARIABLE X(J)      WERT
        4               0.25
        5               0.5
        2               0.5
BASIS VOR ITERATION 3
     VARIABLE X(J)      WERT
        4               0.2
        3               0.4
        2               1
LOESUNG - PRIMALPROBLEM
     VARIABLE X(J)      WERT
        1               0.4
        3               0.16
        2               0.6
SCHATTENPREISE (LOESUNG - DUALPROBLEM)
     VARIABLE X(J)      WERT
        3               0
        4               8
        5              -5.8

     ZIELFUNKTIONSWERT = 3.4 NACH 3 ITERAT.


TABLEAU NACH 3 ITERATIONEN, BASISVARIABLE IN SPALTE 1
        1               1               0               0               2               -.2             0               0.4
        3               0               0               1              -1.2             0.92           -1              0.16
        2               0               1               0              -2               1.2             0               0.6
        0               0               0               0               8              -5.8             0              -3.4
        0               0               0               0               0               1               1               0
```

```
PROBLEM NR. 2
**************

PROBLEMART  MINIMUMAUFGABE

    ZAHL DER ENTSCHEIDUNGSVARIABLEN     2
    ZAHL DER NEBENBEDINGUNGEN           3
             DAVON LE-TYP                       1
             DAVON EQ-TYP                       1
             DAVON GE-TYP                       1

BEISPIEL HAUPT/LOHSE OHNE TABLEAU- UND ZWISCHENSCHRITTAUSDRUCK

VARIABLENBEZEICHNUNG          NUMMERN
ENTSCHEIDUNGSVARIABLE         1  -    2
     NEGATIVE SCHLUPFVARIABLE 3
     POSITIVE SCHLUPFVARIABLE 4
     KUENSTLICHE VARIABLE     5  -          6

LOESUNG - PRIMALPROBLEM
     VARIABLE X(J)      WERT
        1               0.4
        3               0.16
        2               0.6

     ZIELFUNKTIONSWERT = 3.4 NACH 3 ITERAT.



PROBLEM NR. 3
**************

PROBLEMART  MAXIMUMAUFGABE

    ZAHL DER ENTSCHEIDUNGSVARIABLEN     4
    ZAHL DER NEBENBEDINGUNGEN           4
             DAVON LE-TYP                       2
             DAVON EQ-TYP                       2
             DAVON GE-TYP                       0

BEISPIEL MIT RESTRIKTIONEN VOM LE- UND EQ-TYP

VARIABLENBEZEICHNUNG          NUMMERN
ENTSCHEIDUNGSVARIABLE         1  -    4
     POSITIVE SCHLUPFVARIABLE 5  -          6
     KUENSTLICHE VARIABLE     7  -          8
             VARIABLE IN ENTSPRECHENDER REIHENFOLGE IN TABLEAUS,
             AB SPALTE 2, OHNE LETZTE SPALTE (B-VEKT./PRIMALLOES.)

BASIS VOR ITERATION 1
     VARIABLE X(J)      WERT
        5               6
        6               4
        7               0
        8               0
BASIS VOR ITERATION 2
     VARIABLE X(J)      WERT
        5               6
        6               4
        7               0
        4               0
LOESUNG - PRIMALPROBLEM
     VARIABLE X(J)      WERT
        5               3.81818
        3              0.727273
        7               0
        4              0.363636
SCHATTENPREISE (LOESUNG - DUALPROBLEM)
     VARIABLE X(J)      WERT
        5               0
        6               1.09091
        7               0
        8              0.227273

     ZIELFUNKTIONSWERT = 4.36364 NACH 2 ITERAT.
```

Die Dialogversion des vorstehenden Programms ist in einigen Punkten ergänzt. Zunächst ist das Datenbeispiel aus Haupt/Lohse fest eingebaut; es macht den Programmanwender mit der korrekten Dateneingabe vertraut. Dieses Datenbeispiel kann im Dialog beliebig oft wiederholt werden. Bei Verwendung eigener Daten wird schrittweise gefragt, ob die Dateneingabe korrekt war, um sie nötigenfalls wiederholen zu können. Hierdurch werden die bei grösseren Problemen oft auftretenden Eingabefehler sofort korrigierbar gemacht, ohne dass die Dateneingabe für das Gesamtproblem wiederholt werden muss. Nach dem Durchrechnen eines Problems kann vom Anwender jederzeit wieder auf das eingebaute Datenbeispiel zurückgegriffen werden, um eventuelle Anwendungsschwierigkeiten zu beheben.

```
1000 REM     LINEARE PROGRAMMIERUNG - ZWEI-PHASEN METHODE
1005 !
1010 !   URSPRUNGSVERSION DARTMOUTH PUBLIC FILES/MICHIGAN STATE UNIVERSITY,
1015 !   MODIFIZIERT
1020 !
1025 !       NOMENKLATUR
1030 !       Z  =  PROBLEMART, MAX=1, MIN=-1
1035 !       T  =  TABELLENDRUCK, JA=9, NEIN=-9
1040 !       N  =  ZAHL DER ENTSCHEIDUNGSVARIABLEN (STRUKTURVARIABLEN)
1045 !       M  =  ZAHL DER NEBENBEDINGUNGEN
1050 !       L  =  ZAHL DER NEBENBEDINGUNGEN MIT  , ZAHL DER POSITIVEN
1055 !             SCHLUPFVARIABLEN
1060 !       E  =  ZAHL DER NEBENBEDINGUNGEN MIT =
1065 !       G  =  ZAHL DER NEBENBEDINGUNGEN MIT Ö, ZAHL DER NEGATIVEN
1070 !             SCHLUPFVARIABLEN (UEBERSCHUSSVARIABLEN)
1075 !       H  =  E + G = ZAHL DER KUENSTLICHEN VARIABLEN
1080 !       B  =  M + N + G + 1
1085 !       W  =  M
1090 !       (B+1)*(W+2) = SPALTEN + ZEILEN = ERFORDERLICHE GROESSE VON A
1095 !       A  =  SIMPLEXTABLEAU
1100 !
1105 !       DATENEINGABE
1110 !       RESTRIKTIONEN IN REIHENFOLGE DER TYPEN LE, EQ, >= IN DAS
1115 !       AUSGANGSTABLEAU UEBERNEHMEN
1120 !       A) PARAMETER Z BIS G  - PROBLEMSTRUKTUR
1125 !       B) KOEFFIZIENTEN
1130 !            1. RESTRIKTIONSKOEFFIZIENTEN, ZEILENWEISE IN DER FOLGE
1135 !               DER RESTRIKTIONEN VOM TYP LE, EQ, GE. KEINE ANGABE
1140 !               DER SCHLUPFVARIABLEN, KUENSTLICHEN VARIABLEN
1145 !               ODER HILFSVARIABLEN.
1150 !            2. BESCHRAENKUNGSWERTE IN DER REIHENFOLGE DER ZEILEN
1155 !               VON 1.
1160 !            3. ZIELFUNKTIONSKOEFFIZIENTEN
1165 !       C) PROBLEMUMSCHREIBUNG, STANDARDISIERTE STRINGLAENGE 70 ZEICHEN,
1170 !          STRINGKLAMMERN IN SP. 1 UND 70 ODER ENTSPRECHEND VERSCHOBEN
1175 !
1180 REM*****PROBLEMSTRUKTUR**********************************************
1185 LET Q=99999
1190 LET A7=15
1195 LET A8=16
1200 REM A7 UND A8 GEBEN DIE MAXIMALE ZEILEN- UND SPALTENZAHL VON MAT A AN
1205 !   R$ TEXTVARIABLE ZUR PROBLEMUMSCHREIBUNG, 70 ZEICHEN
1210 DIM A(15,16),B$(3)*5,R$*70
1215 LET B$(2)="MUMAU"
1220 LET B$(3)="FGABE"
1225 GOTO  2285
```

```
1230 PRINT "ZAHL DER ENTSCHEIDUNGSVARIABLEN EINGEBEN"
1235 READ N
1240 GOSUB  2395
1245 ON I GOTO  1250, 1230
1250 PRINT "ZAHL DER NEBENBEDINGUNGEN EINGEBEN"
1255 READ M
1260 GOSUB  2395
1265 ON I GOTO  1270, 1250
1270 PRINT "ZAHL DER NEBENBEDINGUNGEN MIT < EINGEBEN"
1275 READ L
1280 GOSUB  2395
1285 ON I GOTO  1290, 1270
1290 PRINT "ZAHL DER NEBENBEDINGUNGEN MIT = EINGEBEN"
1295 READ E
1300 GOSUB  2395
1305 ON I GOTO  1310, 1290
1310 PRINT "ZAHL DER NEBENBEDINGUNGEN MIT > EINGEBEN"
1315 READ G
1320 GOSUB  2395
1325 ON I GOTO  1330, 1310
1330 PAGE
1335 PRINT "PROBLEM NR.";P1
1340 PRINT "***************"
1345 PRINT
1350 PRINT "PROBLEMART ";B$(1);B$(2);B$(3)
1355 PRINT
1360 PRINT "     ZAHL DER ENTSCHEIDUNGSVARIABLEN", TAB (41);N
1365 PRINT "     ZAHL DER NEBENBEDINGUNGEN", TAB (41);M
1370 PRINT "              DAVON LE-TYP", TAB (50);L
1375 PRINT "              DAVON EQ-TYP", TAB (50);E
1380 PRINT "              DAVON GE-TYP", TAB (50);G
1385 PRINT
1390 LET Z=-Z
1395 IF L+E+G=M GOTO  1410
1400 PRINT "PROGRAMMABBRUCH - BEDINGUNG M = L+E+G NICHT EINGEHALTEN"
1405 GOTO  2285
1410 LET B=N+M+G+1
1415 LET W=M
1420 MAT A=ZER(W+2,B+1)
1425 IF (B+1)*(W+2)<A7*A8 GOTO  1435
1430 PRINT "PROGRAMMABBRUCH - PROBLEM ZU GROSS"
1435 IF B+1>A8 GOTO  1445
1440 IF W+2<A7 GOTO  1465
1445 PRINT "PROGRAMMABBRUCH - DIM-STATEMENTAENDERUNG AUF A(";W+2;",";B+1;
1450 PRINT ") NOETIG"
1455 GOTO  2285
1460 REM*****PHASE I*************************************************************
1465 LET M=M-1! INITIALISIERUNG
1470 LET H=1
1475 PRINT "KOEFFIZIENTENMATRIX DER NEBENBEDINGUNGEN EINGEBEN"
1485 FOR I=0 TO M
1490  FOR J=1 TO N
1495   PRINT "KOEFFIZIENT ZEILE=";I+1,"SPALTE=";J
1500   READ A(I+1,J+1)
1505  NEXT J
1510 NEXT I
1515 GOSUB  2395
1520 ON I GOTO  1525, 1475
1525 PRINT "BESCHRAENKUNGSSPALTENVEKTOR EINGEBEN"
1530 FOR I=0 TO M
1535  PRINT "BESCHRAENKUNGSWERT FUER ZEILE=";I+1;" EINGEBEN"
1540  READ A(I+1,B+1)
1545 NEXT I
1550 GOSUB  2395
1555 ON I GOTO  1560, 1525
1560 PRINT "KOEFFIZIENTEN DER ZIELFUNKTION EINGEBEN"
1565 FOR J=1 TO N
1570  PRINT "ZIELFUNKTIONSKOEFFIZIENT SPALTE=";J;" EINGEBEN"
1575  READ A(W+1,J+1)
1580  LET A(W+1,J+1)=Z*A(W+1,J+1)
1585 NEXT J
1590 GOSUB  2395
1595 ON I GOTO  1600, 1560
1600 FOR K=1 TO M+1! TABLEAUAUFBAU
1605  LET A(K,N+G+K+1)=1
1610  LET A(K,1)=K+N+G
```

```
1615 NEXT K
1620 IF G<>0 GOTO  1635
1625 IF E=0 GOTO  1731
1630 GOTO  1650
1635 FOR K=L+E+1 TO M+1
1640  LET A(K,K+N-L-E+1)=-1
1645 NEXT K
1650 LET W=W+1
1655 LET Q=0
1660 FOR J=1 TO N+G
1665  LET S=0
1670  FOR I=M-G-E+1 TO M
1675   LET S=S+A(I+1,J+1)
1680  NEXT I
1685  LET A(W+1,J+1)=-S
1690  IF A(W+1,J+1)>Q GOTO  1705
1695  LET Q=A(W+1,J+1)
1700  LET C=J
1705 NEXT J
1710 LET S=0
1715 FOR J=M-G-E+1 TO M
1720  LET S=S+A(J+1,B+1)
1725 NEXT J
1730 LET A(W+1,B+1)=-S
1731 PRINT "PROBLEMBESCHREIBUNG EINGEBEN MAX. 70 ZEICHEN "
1735 READ R$
1736 GOSUB 2395
1737 ON I GOTO 1740,1731
1740 PRINT R$
1745 PRINT
1750 PRINT "VARIABLENBEZEICHNUNG", TAB (31),"NUMMERN"
1755 PRINT "ENTSCHEIDUNGSVARIABLE"; TAB (31);H; TAB (34);"-"; TAB (39);N
1760 IF G=0 GOTO  1790
1765 IF G>1 GOTO  1780
1770 PRINT "      NEGATIVE SCHLUPFVARIABLE", TAB (31);N+1
1775 GOTO  1790
1780 PRINT "      NEGATIVE SCHLUPFVARIABLE"; TAB (31);N+1;
1785 PRINT TAB (34);"-"; TAB (39);N+G
1790 IF L=0 GOTO  1820
1795 IF L>1 GOTO  1810
1800 PRINT "      POSITIVE SCHLUPFVARIABLE", TAB (31);N+G+1
1805 GOTO  1820
1810 PRINT "      POSITIVE SCHLUPFVARIABLE"; TAB (31);N+G+1;
1815 PRINT TAB (34);"-"; TAB (39);N+G+L
1820 IF G+E=0 GOTO  1850
1825 IF G+E>1 GOTO  1840
1830 PRINT "      KUENSTLICHE VARIABLE", TAB (31);N+G+L+1
1835 GOTO  1850
1840 PRINT "      KUENSTLICHE VARIABLE"; TAB (31);N+G+L+1;
1845 PRINT TAB (34);"-"; TAB (39);B-1
1850 IF T=-9 GOTO  1870
1855 PRINT "              VARIABLE IN ENTSPRECHENDER REIHENFOLGE IN TABLEAUS,"
1860 PRINT "              AB SPALTE 2, OHNE LETZTE SPALTE (B-VEKT./PRIMALLOES.)"
1865 GOSUB  2365
1870 PRINT ! PIVOTIERUNG
1875 IF Q=99999 GOTO  2045
1880 IF Q=0 GOTO  2085
1885 GOTO  2150
1890 LET H=H+1
1895 LET Q=10**37
1900 LET R=-1
1905 FOR I=0 TO M
1910  IF A(I+1,C+1)<=0 GOTO  1930
1915  IF A(I+1,B+1)/A(I+1,C+1)>Q GOTO  1930
1920  LET Q=A(I+1,B+1)/A(I+1,C+1)
1925  LET R=I
1930 NEXT I
1935 IF R>=-0.5 GOTO  1955
1940 PRINT "LOESUNG UNBEGRENZT"
1945 IF T=-9 GOTO  2285
1950 GOSUB  2365
1955 LET P=A(R+1,C+1)
1960 LET A(R+1,1)=C
1965 FOR J=1 TO B
1970  LET A(R+1,J+1)=A(R+1,J+1)/P
1975 NEXT J
```

```
FOR I=0 TO W
 IF I=R GOTO  2020
 FOR J=1 TO B
  IF J=C GOTO  2015
  LET A(I+1,J+1)=A(I+1,J+1)-A(R+1,J+1)*A(I+1,C+1)
  IF ABS(A(I+1,J+1))>0.00001 GOTO  2015
  LET A(I+1,J+1)=0
 NEXT J
NEXT I
FOR I=0 TO W
 LET A(I+1,C+1)=0
NEXT I
LET A(R+1,C+1)=1
LET Q=0
FOR J=1 TO N+G+L
 IF A(W+1,J+1)>Q GOTO  2070
 LET Q=A(W+1,J+1)
 LET C=J
NEXT J
GOTO  1880
REM*****PHASE II**********************************************************************
IF W=M+1 GOTO  2145  ! LOESBARKEITSTEST
LET W=W-1
IF A(W+2,B+1)<0.000001 GOTO  2110
PRINT "KEINE ZULAESSIGE LOESUNG"
GOTO  2285
FOR I=0 TO M
 IF A(I+1,1)<=N+G+L GOTO  2135
 FOR J=1 TO B
  LET A(I+1,J+1)=0
 NEXT J
NEXT I
GOTO  2045
PRINT "LOESUNG - PRIMALPROBLEM"! PIVOTIERUNG/ERGEBNISAUSDRUCK
IF Q=0 GOTO  2165
IF T=-9 GOTO  2190
PRINT "BASIS VOR ITERATION";H
PRINT "     VARIABLE X(J)", TAB (24);"WERT"
FOR I=0 TO M
 IF A(I+1,1)=0 GOTO  2185
 PRINT A(I+1,1),A(I+1,B+1)
NEXT I
IF Q<>0 GOTO  1890
IF T=-9 GOTO  2225
PRINT "SCHATTENPREISE (LOESUNG - DUALPROBLEM)"
PRINT "     VARIABLE X(J)", TAB (24);"WERT"
FOR J=N+1 TO B-G-1
 PRINT J,A(W+1,J+1)
NEXT J
PRINT
PRINT "     ZIELFUNKTIONSWERT =";-Z*A(W+1,B+1);"NACH";H-1;"ITERAT."
PRINT
IF T=-9 GOTO  2250
GOSUB  2365
PRINT "NEUES PROBLEM,JA=J, NEIN=N BEISPIELPROBLEM WIEDERHOLEN=W"
INPUT A$
IF A$="J" GOTO  2285
IF A$="N" GOTO  2595
IF A$="W" GOTO  2280
GOTO  2250
RESTORE
PRINT "BREITE DER AUSGABENZEILE IN ZEICHEN EINGEBEN"
READ Z
GOSUB 2395
ON I GOTO 2289,2285
Z=INT(Z/15)
IF Z>8 LET Z=8
WIDTH Z
PRINT "PROBLEMART EINGEBEN, MAX=1  MIN=-1"
READ Z
GOSUB  2395
ON I GOTO  2305, 2285
PRINT "TABELLEN DRUCKEN ? JA=9 NEIN=-9"
READ T
GOSUB  2395
ON I GOTO  2325, 2305
```

```
2325 LET P1=P1+1
2330 IF Z=1 GOTO  2345
2335 LET B$(1)=" MINI"
2340 GOTO  2350
2345 LET B$(1)=" MAXI"
2350 IF ABS(Z)+ABS(T)=10 GOTO  1230
2355 PRINT "DATENEINGABE KONTROLLIEREN"
2360 GOTO  2250
2365 REM TABLEAUDRUCK
2370 IF B>7 GOTO  2390
2375 PRINT
2380 PRINT "TABLEAU NACH";H-1;"ITERATIONEN, BASISVARIABLE IN SPALTE 1"
2385 MAT PRINTA,
2390 RETURN
2395 REM EINGABE PRUEFEN
2400 LOOP
2405  I=0
2410  PRINT "EINGABE SOWEIT OKAY J ODER N ? "
2415  READ A$
2420  SELECT A$
2425 CASE IS="J"
2430   I=1
2435 CASE IS = "N"
2440   I=2
2445  ENDSELECT
2450 ENDLOOPUNTIL I<>0
2455 RETURN
2460 !
2465 ! BEISPIEL EINER DATENEINGABE FUER EIN LP-PROBLEM
2470 !
2471 ! ZEILENBREITE = 80 ZEICHEN
2472 DATA 80,"J"
2473 !
2475 ! PARAMETER  Z   T   N   M  DER PROBLEMSTRUKTUR
2480 DATA  -1,"J",  9,"J",  2,"J",  3,"J"
2485 !
2490 ! NEBENBEDINGUNGEN <=  =  >=
2495 DATA          1,"J",  1,"J",  1,"J"
2500 !
2505 ! KOEFFIZIENTEN DER RESTRIKTIONEN
2510 DATA  0.6,  0.1       ! LE-TYP
2515 DATA  1.0,  1.0       ! EQ-TYP
2520 DATA  0.2,  0.8       ! GE-TYP
2525 DATA "J"
2530 !
2535 ! BESCHRAENKUNGSVEKTOR
2540 DATA 0.3, 1.0, 0.4
2545 DATA "J"
2550 !
2555 ! KOEFFIZIENTEN DER ZIELFUNKTION
2560 DATA 1.0, 5.0, "J"
2565 !
2570 ! PROBLEMUMSCHREIBUNG
2575 DATA "AUFGABE AUS HAUPT/LOHSE, GRUNDLAGEN UND ANWENDUNG DER LIN.O OPT."
2576 DATA "J"
2580 !
2585 ! ENDE DER DATENEINGABE FUER EIN LINEARES OPTIMIERUNGSPROBLEM
2590 !
2595 END
```

```
FOSBIC-2 FOSBIC-2 FOSBIC-2 FOSBIC-2 FOSBIC-2  ** VERSION 2.3 ** APRIL 1983
**********************************************************************

NON PRINT


BREITE DER AUSGABENZEILE IN ZEICHEN EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
PROBLEMART EINGEBEN, MAX=1  MIN=-1
EINGABE SOWEIT OKAY J ODER N ?
TABELLEN DRUCKEN ? JA=9 NEIN=-9
EINGABE SOWEIT OKAY J ODER N ?
ZAHL DER ENTSCHEIDUNGSVARIABLEN EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
ZAHL DER NEBENBEDINGUNGEN EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
ZAHL DER NEBENBEDINGUNGEN MIT < EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
ZAHL DER NEBENBEDINGUNGEN MIT = EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
ZAHL DER NEBENBEDINGUNGEN MIT > EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?



PROBLEM NR. 1
**************

PROBLEMART  MINIMUMAUFGABE

    ZAHL DER ENTSCHEIDUNGSVARIABLEN      2
    ZAHL DER NEBENBEDINGUNGEN            3
            DAVON LE-TYP                         1
            DAVON EQ-TYP                         1
            DAVON GE-TYP                         1

KOEFFIZIENTENMATRIX DER NEBENBEDINGUNGEN EINGEBEN
KOEFFIZIENT ZEILE= 1          SPALTE= 1
KOEFFIZIENT ZEILE= 1          SPALTE= 2
KOEFFIZIENT ZEILE= 2          SPALTE= 1
KOEFFIZIENT ZEILE= 2          SPALTE= 2
KOEFFIZIENT ZEILE= 3          SPALTE= 1
KOEFFIZIENT ZEILE= 3          SPALTE= 2
EINGABE SOWEIT OKAY J ODER N ?
BESCHRAENKUNGSSPALTENVEKTOR EINGEBEN
BESCHRAENKUNGSWERT FUER ZEILE= 1  EINGEBEN
BESCHRAENKUNGSWERT FUER ZEILE= 2  EINGEBEN
BESCHRAENKUNGSWERT FUER ZEILE= 3  EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
KOEFFIZIENTEN DER ZIELFUNKTION EINGEBEN
ZIELFUNKTIONSKOEFFIZIENT SPALTE= 1  EINGEBEN
ZIELFUNKTIONSKOEFFIZIENT SPALTE= 2  EINGEBEN
EINGABE SOWEIT OKAY J ODER N ?
PROBLEMBESCHREIBUNG EINGEBEN MAX. 70 ZEICHEN
EINGABE SOWEIT OKAY J ODER N ?
AUFGABE AUS HAUPT/LOHSE, GRUNDLAGEN UND ANWENDUNG DER LIN.O OPT.

VARIABLENBEZEICHNUNG          NUMMERN
ENTSCHEIDUNGSVARIABLE         1  -    2
    NEGATIVE SCHLUPFVARIABLE 3
    POSITIVE SCHLUPFVARIABLE 4
    KUENSTLICHE VARIABLE      5  -    6
          VARIABLE IN ENTSPRECHENDER REIHENFOLGE IN TABLEAUS,
          AB SPALTE 2, OHNE LETZTE SPALTE (B-VEKT./PRIMALLOES.)

TABLEAU NACH 0 ITERATIONEN, BASISVARIABLE IN SPALTE 1
      4              0.6             0.1              0                1
      0              0               0.3
      5              1               1                0                0
      1              0               1
      6              0.2             0.8             -1                0
      0              1               0.4
      0              1               5                0                0
      0              0               0
      0             -1.2            -1.8              1                0
      0              0              -1.4
```

```
BASIS VOR ITERATION 1
     VARIABLE X(J)      WERT
        4               0.3
        5               1
        6               0.4
BASIS VOR ITERATION 2
     VARIABLE X(J)      WERT
        4               0.25
        5               0.5
        2               0.5
BASIS VOR ITERATION 3
     VARIABLE X(J)      WERT
        4               0.2
        3               0.4
        2               1
LOESUNG - PRIMALPROBLEM
     VARIABLE X(J)      WERT
        1               0.4
        3               0.16
        2               0.6
SCHATTENPREISE (LOESUNG - DUALPROBLEM)
     VARIABLE X(J)      WERT
        3               0
        4               8
        5              -5.8

     ZIELFUNKTIONSWERT = 3.4 NACH 3 ITERAT.


TABLEAU NACH 3 ITERATIONEN, BASISVARIABLE IN SPALTE 1
        1               1               0               0               2
        -.2             0               0.4
        3               0               0               1              -1.2
        0.92           -1               0.16
        2               0               1               0              -2
        1.2             0               0.6
        0               0               0               0               8
       -5.8             0              -3.4
        0               0               0               0               0
        1               1               0
NEUES PROBLEM,JA=J, NEIN=N BEISPIELPROBLEM WIEDERHOLEN=W
```

53 Volkswirtschaftslehre

531 Input-Output Analyse

Wassily Leontief und Léon Walras begründeten als erste für die Volkswirtschaftslehre die Theorie der Input-Output Analyse[29], wobei Leontiefs Werk[30] als „natürliche und unvermeidliche Folge von Walras' allgemeiner Theorie" erscheint[31]. Die Input-Output Theorie ist deshalb heute von fundamentaler Bedeutung für die moderne Wirtschaftswissenschaft und "can be considered [as] the forerunner of the linear programming theory"[32].

Trotz der strengen Annahmen[33] der
- Linearität
- Homogenität und
- Nichtsubstitutionsfähigkeit

der Produktionsfunktionen und der Probleme der Klassifikation und Aggregation[34] wird die Input-Output Theorie zur Analyse komplexer mikro- und makroökonomischer Prozesse und Systeme angewendet[35].

Das volkswirtschaftliche Beispiel zur Input-Output Analyse von Heinz Stöwe und Erich Härtter[36] ist mit dem im Abschnitt 512 gezeigten BASIC-Programm durchgerechnet worden, wobei folgende Tableaus ausgedruckt wurden.

29 Mattessich, Die wissenschaftlichen Grundlagen des Rechnungswesens, S. 222; Farag, Input-Output Analysis, p. 28–31 and p. 37–39.

30 Wassily Leontief. The Structure of American Economy, 1919–1929. Cambridge, Massachusetts 1941; Wassily Leontief. The Structure of American Economy, 1919–1939. New York 1951.

31 Mattessich, Die wissenschaftlichen Grundlagen des Rechnungswesens, S. 186.

32 Chenery B. Hollis. The Use of Interindustry Analysis in Development Programming. In: Tibor Barna (ed.), Structural Interdependence and Economic Development, New York 1963, p. 12.

33 Mattessich, Die wissenschaftlichen Grundlagen des Rechnungswesens, S. 225; Farag, Input-Output Analysis, p. 34–36.

34 Farag, Input-Output Analysis, p. 42.

35 Jonas Kornai. Mathematische Methoden bei der Planung der ökonomischen Struktur. Budapest 1967, S. 50–61, S. 221–224 und S. 296–297; Farag, Input-Output Analysis, p. 42–49.

36 Stöwe und Härtter, Lehrbuch der Mathematik für Volks- und Betriebswirte, S. 220–223.

```
LIST OF DATA AFTER END STATEMENT
"--------- VOLKSWIRTSCHAFTLICHES BEISPIEL ---------"
3,3,2
4,26,1
1,41,5
3,12,6
10,1,6
45,7,46
27,13,4
10,8,1
30,58,52
"SEKTOR-1","SEKTOR-2","SEKTOR-3"
"HAUSHALTE","   STAAT","INV+LAG+EX"
"IMPORT","SONSTIGES"
10,1,6.59
45,7,48
27,13,4.35
"ENDE"
*
```

STARTTABLEAU

--------- VOLKSWIRTSCHAFTLICHES BEISPIEL ---------

			----OUTPUT----				
----INPUT----	SEKTOR-1	SEKTOR-2	SEKTOR-3	HAUSHALTE	STAAT	INV+LAG+EX	TOTAL
SEKTOR-1	4	26	1	10	1	6	48
SEKTOR-2	1	41	5	45	7	46	145
SEKTOR-3	3	12	6	27	13	4	65
IMPORT	10	8	1				19
SONSTIGES	30	58	52				140
TOTAL	48	145	65	82	21	56	417

ENDTABLEAU

--------- VOLKSWIRTSCHAFTLICHES BEISPIEL ---------

			----OUTPUT----				
----INPUT----	SEKTOR-1	SEKTOR-2	SEKTOR-3	HAUSHALTE	STAAT	INV+LAG+EX	TOTAL
SEKTOR-1	4.10198	26.5205	1.0113	10	1	6.59	49.2238
SEKTOR-2	1.0255	41.8208	5.0565	45	7	48	147.903
SEKTOR-3	3.07649	12.2402	6.0678	27	13	4.35	65.7345
IMPORT	10.255	8.16015	1.0113				19.4264
SONSTIGES	30.7649	59.1611	52.5876				142.514
TOTAL	49.2238	147.903	65.7345	82	21	58.94	424.801

In einer Volkswirtschaft sollen n Güter in m Produktionsprozessen hergestellt werden[37]. Die produzierten Güter gehen dabei auch als Produktionsfaktoren in die Prozesse ein. Für die Produktionsprozesse werden einfache Leontief-Produktionsfunktionen angenommen, womit jeder Prozess durch konstante technische Koeffizienten beschrieben werden kann. Enthält die Matrix A

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

mit den Feldgrenzen (M,N) die technischen Input-Koeffizienten, die Matrix B

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ \vdots & & & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix}$$

mit den Feldgrenzen (M,N) die technischen Output-Koeffizienten und der Zeilenvektor

$$1 = [1_1 1_2 \dots 1_m]$$

mit den Feldgrenzen (M) die Prozessintensitäten, so ergibt sich aus der Matrizenmultiplikation

$$x = 1 \cdot A$$

der Zeilenvektor x der Inputmengen mit der Feldgrenze (N) und aus der Matrizenoperation

$$z = 1 \cdot B$$

der Zeilenvektor z der Outputmengen mit der gleichen Feldgrenze. Analog zu dem obigen Gütertransformationssystem wird ein monetäres Kosten- und Umsatzsystem definiert, was sich aus den Matrizengleichungen

$$k = A \cdot p$$

37 Vgl. zu diesem Abschnitt: Kurt Elsner. Wachstums- und Konjunkturtheorie. In: Kompendium der Volkswirtschaftslehre Bd. 1, 2. Aufl., Göttingen 1969, S. 294–299.

und

$$u = B \cdot q$$

aufbaut, wobei k der Kostenspaltenvektor mit der Feldgrenze (M,1) und u der Umsatzspaltenvektor mit der gleichen Feldgrenze ist. p wird als Preisspaltenvektor (N,1) für die Preise am Anfang der Periode t und q als Preisspaltenvektor für die Preise am Ende der Periode t definiert. Die Differenz zwischen den Vektoren u und k

$$GuV = u - k$$

stellt den Gewinn oder Verlust der einzelnen Produktionsprozesse dar. Die Wachstumsfaktoren w_i der n Güter ergeben sich aus der Division der Outputmenge z_i durch die Inputmenge x_i

$$w_i = \frac{z_i}{x_i} \qquad i = 1(1)N$$

Ebenso errechnet sich der interne Zinsfaktor c_i eines Produktionsprozesses aus der Gleichung:

$$c_i = \frac{u_i}{k_i} \qquad i = 1(1)M$$

Der minimale Wachstumsfaktor wird mit W1 und der maximale Zinsfaktor mit C1 bezeichnet.
Im Gegensatz zu den von Neumann Postulaten[38] wird angenommen, dass bei der Bedingung

$$W1 \cdot x_i \geq z_i$$

der Preis für die Periode $t + 1$ sich nach der Beziehung

$$q_{i,t+1} = q_{i,t} + q_i \cdot rnd \quad i = 1(1)N$$

und für die Bedingung

$$W1 \cdot x_i < z_i$$

aus der Gleichung

$$q_{i,t+1} = q_{i,t} \cdot rnd \quad i = 1(1)N$$

berechnet, wobei rnd eine Zufallszahl aus einer gleichverteilten Zufallszahlenreihe zwischen 0 und 1 darstellt. Für die Prozessintensitäten l_i wird

38 Elsner, Wachstums- und Konjunkturtheorie, S. 297–298.

analog verfahren und bei der Bedingung

$$C1 \cdot k_i \leq u_i$$

die Prozessintensitäten für die Folgeperiode nach

$$l_{i,t+1} = l_{i,t} + l_{i,t} \cdot \text{rnd} \quad i = 1(1)M$$

und bei Eintreffen der Bedingung

$$C1 \cdot k_i > u_i$$

nach

$$l_{i,t+1} = l_{i,t} \cdot \text{rnd} \quad i = 1(1)M$$

berechnet. rnd ist hierbei wiederum eine Zufallszahl mit Gleichverteilung zwischen 0 und 1.
Auf Grund der obigen Annahme, dass die Outputmengen der Periode t als Inputmengen für die Periode t + 1 verwendet werden, kann sich eine Differenz zwischen Bedarf in der Periode t + 1 und der Produktion aus Periode t ergeben, der durch den Zeilenvektor r der Lagermengenänderung mit der Feldgrenze (N) ausgedrückt wird. Der jeweilige Zeilenvektor der Lagermengen ergibt sich aus der Gleichung

$$s_t = s_{t-1} + r_t$$

die strenge Annahme des von Neumann'schen Modells, dass die Outputmengen der Periode t vollständig als Inputmengen der Periode t + 1 verwendet werden, entfällt[39].
Im folgenden BASIC-Programm ist das obige multisektorale Spielmodell programmiert. Die einfache Modellstruktur auf der Basis von Matrizengleichungen erlaubt es dem Benutzer, leicht Variationen vorzunehmen. Zum Beispiel können die Änderungen der Prozessintensitäten am Ende jeder Periode von den Lagermengen bzw. von den Lagermengenänderungen funktionell abhängig gemacht werden, so dass folgende Funktionen zu programmieren sind:
Bei Eintreffen der Bedingung

$$C1 \cdot k_i \leqq u_i$$

die Gleichung

$$l_{i,t+1} = l_{i,t} + \frac{r_t \cdot \text{rnd}}{100} \qquad i = 1(1)M$$

39 Elsner, Wachstums- und Konjunkturtheorie, S. 295–296.

bzw. bei Nichteintreffen der Bedingung die Beziehung

$$l_{i,t+1} = l_{i,t} - \frac{r_t \cdot rnd}{100} \qquad i = 1(1)M$$

Ebenso bietet es sich an, die Gewinne bzw. Verluste der Periode t in die Berechnung der Preise für die Periode t + 1 einzubeziehen.
Die Unabhängigkeit der technischen Koeffizienten von der Zeit stellt ebenfalls nur eine Annahme im obigen Spielmodell dar und kann vom Anwender leicht abgeändert werden.
Die Beispieldaten sind willkürlich gewählt. Die Ergebnisse weichen trotz gleicher Anfangsdaten von denjenigen der Erstausgabe (S. 208–209) ab. Die Ursache dafür liegt in der andersartigen Zufallszahlenfolge des verwendeten Computersystems (CD Cyber 174).

```
1000 DIM A(5,6),B(5,6),L(5),X(6),Z(6),K(5,1),U(5,1),P(6,1),Q(6,1)
1005 DIM C(5),W(6),R(6),S(6)
1010 READ M,N,N9
1015 DATA 4,5,10
1020 MAT A=ZER(M,N)
1025 MAT B=ZER(M,N)
1030 MAT L=ZER(M)
1035 MAT X=ZER(N)
1040 MAT R=ZER(N)
1045 MAT S=ZER(N)
1050 MAT Z=ZER(N)
1055 MAT K=ZER(M,1)
1060 MAT U=ZER(M,1)
1065 MAT P=ZER(N,1)
1070 MAT C=ZER(M)
1075 MAT W=ZER(N)
1080 MAT Q=ZER(N,1)
1085 MAT READ A,B,L,P
1090 PRINT "MATRIX DER TECHNISCHEN INPUT-KOEFFIZIENTEN"
1095 MAT PRINT A
1100 PRINT "MATRIX DER TECHNISCHEN OUTPUT-KOEFFIZIENTEN"
1105 MAT PRINT B
1110 LET T=1
1115 MAT X=L*A
1120 PRINT "VEKTOR DER PROZESSINTENSITAETEN"
1125 MAT PRINT L
1130 PRINT "PREISE IN DER PERIODE ";T
1135 MAT PRINT P
1140 PRINT "INPUT-MENGEN"
1145 MAT PRINT X
1150 IF T=1 GOTO  1185
1155 MAT R=Z-X
1160 MAT S=S+R
1165 PRINT "VEKTOR DER LAGERAENDERUNGEN"
1170 MAT PRINT R
1175 PRINT "VEKTOR DER LAGERMENGEN"
1180 MAT PRINT S
1185 MAT Z=L*B
1190 PRINT "OUTPUT-MENGEN"
1195 MAT PRINT Z
1200 FOR I=1 TO N
1205  LET W(I)=Z(I)/X(I)
1210 NEXT I
1215 LET W1=10**10
1220 FOR I=1 TO N
1225  IF W(I)>W1 GOTO  1235
1230  LET W1=W(I)
1235 NEXT I
1240 IF T>1 GOTO  1250
```

```
 1245 MAT Q=P
 1250 MAT K=A*P
 1255 MAT U=B*Q
 1260 PRINT "    KOSTEN","    UMSATZ","GEWINN/VERLUST"
 1265 FOR I=1 TO M
 1270  PRINT K(I,1),U(I,1),U(I,1)-K(I,1)
 1275 NEXT I
 1280 FOR I=1 TO M
 1285  LET C(I)=U(I,1)/K(I,1)
 1290 NEXT I
 1295 LET C1=0
 1300 FOR I=1 TO M
 1305  IF C(I)<=C1 GOTO  1315
 1310  LET C1=C(I)
 1315 NEXT I
 1320 MAT P=Q
 1325 PRINT "WACHSTUMSVEKTOR"
 1330 MAT PRINT W
 1335 PRINT "ZINSVEKTOR"
 1340 MAT PRINT C
 1345 PRINT "MINIMALE WACHSTUMSRATE =";W1
 1350 PRINT "MAXIMALER ZINSSATZ =";C1
 1355 FOR I=1 TO N
 1360  IF W1*X(I)>=Z(I) GOTO  1375
 1365  LET Q(I,1)=Q(I,1)*RND(1)
 1370  GOTO  1380
 1375  LET Q(I,1)=Q(I,1)+Q(I,1)*RND(1)
 1380 NEXT I
 1385 FOR I=1 TO M
 1390  IF C1*K(I,1)<=U(I,1) GOTO  1405
 1395  LET L(I)=L(I)*RND(1)
 1400  GOTO  1410
 1405  LET L(I)=L(I)+L(I)*RND(1)
 1410 NEXT I
 1415 PRINT "PREISE AM ENDE DER PERIODE ";T
 1420 MAT PRINT Q
 1425 LET T=T+1
 1430 IF T>N9 GOTO  1455
 1435 PAGE
 1440 PRINT "ZEIT =";T
 1445 IF ABS(C1-W1)<0.001 GOTO  1455
 1450 GOTO  1115
 1455 STOP
 1460 DATA 12,15,16,9,37,  40,12,22,38,41,  21,29,37,46,17,  27,44,25,7,5
 1465 DATA 5,6,40,58,25,  23,53,22,6,20,  44,20,24,21,29,  28,21,14,15,26
 1470 DATA 5,12,9,8
 1475 DATA 14,2,6,8,5
 1480 END
MATRIX DER TECHNISCHEN INPUT-KOEFFIZIENTEN
       12             15             16              9             37
       40             12             22             38             41
       21             29             37             46             17
       27             44             25              7              5
MATRIX DER TECHNISCHEN OUTPUT-KOEFFIZIENTEN
        5              6             40             58             25
       23             53             22              6             20
       44             20             24             21             29
       28             21             14             15             26
VEKTOR DER PROZESSINTENSITAETEN
        5             12              9              8
PREISE IN DER PERIODE  1
       14
        2
        6
        8
        5
INPUT-MENGEN
      945            832            877            971            870
OUTPUT-MENGEN
      921           1014            792            671            834
    KOSTEN        UMSATZ  GEWINN/VERLUST
       551            911            360
      1225            708           -517
      1027           1113             86
       697            768             71
```

```
WACHSTUMSVEKTOR
      0.974603        1.21875       0.903079        0.69104       0.958621
ZINSVEKTOR
       1.65336       0.577959        1.08374        1.10187
MINIMALE WACHSTUMSRATE = 0.69104
MAXIMALER ZINSSATZ = 1.65336
PREISE AM ENDE DER PERIODE  1
       11.5518
       0.673602
        5.19099
        0.59069
        2.73725


ZEIT = 2
VEKTOR DER PROZESSINTENSITAETEN
       7.87689       7.83003       6.62893       4.04602
PREISE IN DER PERIODE  2
       14
        2
        6
        8
        5
INPUT-MENGEN
      656.174       582.377       644.712       701.686       745.398
VEKTOR DER LAGERAENDERUNGEN
      264.826       431.623       147.288      -30.6858       88.6022
VEKTOR DER LAGERMENGEN
      264.826       431.623       147.288      -30.6858       88.6022
OUTPUT-MENGEN
      624.436       679.798       703.075       703.737       650.958
    KOSTEN          UMSATZ     GEWINN/VERLUST
      551           372.132      -178.868
     1225           473.884      -751.116
     1027           738.121      -288.879
      697           490.3        -206.7
WACHSTUMSVEKTOR
      0.951633        1.16728       1.09053       1.00292       0.873303
ZINSVEKTOR
       0.675375       0.386844       0.718716       0.703443
MINIMALE WACHSTUMSRATE = 0.873303
MAXIMALER ZINSSATZ = 0.718716
PREISE AM ENDE DER PERIODE  2
       11.4221
       0.579331
       0.693984
       0.370048
        2.27455


ZEIT = 3
VEKTOR DER PROZESSINTENSITAETEN
       2.36937       5.74711       9.66795       1.44786
PREISE IN DER PERIODE  3
       11.5518
       0.673602
        5.19099
        0.59069
        2.73725
INPUT-MENGEN
      500.436       448.582       558.257       694.575       494.893
VEKTOR DER LAGERAENDERUNGEN
      124           231.215       144.818        9.16228      156.066
VEKTOR DER LAGERMENGEN
      388.827       662.838       292.106      -21.5235       244.668
OUTPUT-MENGEN
      609.96        542.577       473.512       396.651       492.191
    KOSTEN          UMSATZ     GEWINN/VERLUST
      338.376       166.672      -171.704
      719.032       356.391      -362.641
      527.895       604.545        76.6507
      489.134       406.388       -82.7459
WACHSTUMSVEKTOR
       1.21886        1.20954       0.848197       0.57107       0.994542
ZINSVEKTOR
       0.492564       0.495653       1.1452        0.830832
```

```
MINIMALE WACHSTUMSRATE = 0.57107
MAXIMALER ZINSSATZ = 1.1452
PREISE AM ENDE DER PERIODE  3
        7.51631
       0.272938
        0.55616
       0.349902
        1.37073
```

533 Modifiziertes Oszillationsmodell nach J. R. Hicks

Bei dem im Band 1 Abschnitt 422 vorgestellten Oszillationsmodell nach J. R. Hicks ist die Akzeleratorhypothese nur sinnvoll, wenn die Kapazitäten stets voll ausgelastet sind. Durch die Einführung der Ober- und Untergrenzen wird diese Vollanpassung verhindert, was zu einem „merkwürdigem Investitionsverhalten" führt, indem die Unternehmen in der 11. und 19. Periode bereits wieder investieren, obwohl die Kapazitäten noch nicht voll ausgelastet sind[40].

Das in diesem Fall unrealistische Akzeleratorprinzip wird deshalb durch die Kapitalanpassungshypothese ersetzt. Hierbei wird angenommen, dass „durch Investitionen die in der betreffenden oder der vorangegangenen Periode aufgetretene Lücke an Sachkapital ganz oder teilweise beseitigt wird"[41].

Es wird nun die Investitionsfunktion

$$I^{ind} = V1\,(Q_{t-1} - L_{t-1})$$

unterstellt, wobei mit Q_{t-1} der erforderliche und mit L_{t-1} der tatsächliche Kapitalbestand in der Periode $t-1$ bezeichnet wird.

Der erforderliche Kapitalbestand 1 wird als proportional zum Volkseinkommen Z angenommen.

$$Q_t = b \cdot Z_t$$

Das modifizierte Hicks-Modell basiert nun auf folgenden Gleichungen[42]:

$K_t = A_o(1+r)^t$	autonome Investitionen
$J_t^{ind} = V1\,(Q_{t-1} - L_{t-1})$	induzierte Investitionen
$J_t^{ind,max} = A_3(1+r_3)^t$	Obergrenze der induzierten Investitionen

40 Jürgen Kromphardt. Wachstum und Konjunktur. Grundlagen einer theoretischen Analyse und wirtschaftspolitischen Steuerung. Göttingen 1972. S. 180–181.

41 Kromphardt, Wachstum und Konjunktur, S. 181.

42 Vgl. hierzu auch: Kromphardt, Wachstum und Konjunktur, S. 182 und S. 185.

$J_t^{ind,min} = A_4(1 + r_4)^t$	Untergrenze der induzierten Investitionen
$I_t = K_t + J_t$	Gesamtinvestitionen
$C_t = C1 \cdot Z_{t-1}$	Konsum
$C_t^{max} = A_1(1 + r_1)^t$	Obergrenze des Konsums
$C_t^{min} = A_2(1 + r_2)^t$	Untergrenze des Konsums
$Z_t = I_t + C_t$	Volkseinkommen
$L_t = L_{t-1} + J_t$	tatsächlicher Kapitalbestand
$Q_t = b \cdot Z_t$	erforderlicher Kapitalbestand

Die Änderungen des BASIC-Programms aus Band 1 Abschnitt 422 zur Darstellung des modifizierten Hicks-Modells betreffen die Eingabe der Startwerte sowie die Modellgleichungen im Unterprogramm, wobei hier nur die Gleichungen für die induzierten Investitionen und für den erforderlichen Kapitalbestand neu eingefügt werden müssen.
Eine zusätzliche Programmvariante wird durch das Zwischenspeichern aller Modellwerte in einer relativ organisierten Datei mit 8 Sätzen gezeigt. Hierdurch entfällt das zweimalige Durchrechnen des Modells, da die Werte der aufzuzeichnenden Modellvariablen nun indexbezogen aus der Datei entnommen werden[43]. Die Ursprungsform der Dateiverarbeitung nach der FOSBIC-Version 1 ist als Kommentar beibehalten; im übrigen entsprechen die Anweisungen der Norm von ANSI BASIC 82.[44]

```
1000 REM MODIFIZIERTES MODELL NACH HICKS
1005 DIM I(3),J(3),K(3),Z(3),L(3),C(3),M$*15,U$*1,V$*1,Z$*1
1010 DIM A1$*20,A2$*20,A3$*20,A4$*20,A5$*20
1015 A1$="DATEINAME"
1020 A2$="RELATIVE"
1025 A3$="INTERNAL"
1030 A4$="VARIABLE"
1035 A5$="INOUT"
1040 OPEN #8,NAME A1$, ORGANIZATION A2$, RECTYPE A3$, RECSIZE A4$,ACCESS A5$
1045 !******** ALTE VERSION ********  FOSBIC VERSION 1 *********
1050 !  OPEN #DATEI#;8
1055 DIM Q(3)
1060 LET U$=" "
1065 LET U$=" "
1070 LET V$="*"
1075 LET Z$="I"
```

43 Beispieldaten entnommen aus: Kromphardt, Wachstum und Konjunktur, S. 185.
44 Vgl. Weber und Türschmann, BASIC 2, 1. Aufl., S. 99–123.

```
1080 REM EINLESEN DER MODELLPARAMETER
1085 READ C1,V1,A0,R,A1,R1,A2,R2,A3,R3,A4,L(2),Q(2),B,Z(1),Z(2),N9,MS
1090 DATA 0.75,1,250,0.03,1000,0.03,500,0.03,1000,0.03,-0.1,3000,3000,3,1000
1095 DATA 1000,3,"GES.INVESTITION"
1100 COMMENT C1 = KONSUMQUOTE
1105 COMMENT V1 = RATE DER INDUZIERTEN INVESTITIONEN
1110 COMMENT A0 UND R FUNKTIONSPARAMETER DER AUTONOMEN INVESTITIONEN
1115 COMMENT A1 UND R1 KONSUMPARAMETER DER OBERGRENZE
1120 COMMENT A2 UND R2 KONSUMPARAMETER DER UNTERGRENZE
1125 COMMENT A3 UND R3 INDUZIERTE INVESTITIONS PARAMETER DER OBERGRENZE
1130 COMMENT A4 INDUZIERTE INVESTITIONS-PARAMETER DER UNTERGRENZE
1135 COMMENT L(2) = ANFANGSKAPITALBESTAND VOR DER PERIODE T
1140 COMMENT Y(2) UND Y(1) SIND VOLKSEINKOMMEN VOR DER PERIODE T
1145 COMMENT Q(2) = TATSAECHLICHER KAPITALBESTAND VOR DER PERIODE T
1150 COMMENT N9 = STEUERPARAMETER ZUR AUSWAHL DER AUFZUZEICHNENDEN VARIABLEN
1155 COMMENT N9=1 ZEICHNET K(T)
1160 COMMENT N9=2 ZEICHNET J(T)
1165 COMMENT N9=3 ZEICHNET I(T)
1170 COMMENT N9=4 ZEICHNET C(T)
1175 COMMENT N9=5 ZEICHNET Z(T)
1180 COMMENT N9=6 ZEICHNET L(T)
1185 COMMENT N9=7 ZEICHNET Q(T)
1190 PRINT "           C1 =",C1,"          V1 =",V1
1195 PRINT "           A0 =",A0,"           R =",R
1200 PRINT "           A1 =",A1,"          R1 =",R1
1205 PRINT "           A2 =",A2,"          R2 =",R2
1210 PRINT "           A3 =",A3,"          R3 =",R3
1215 PRINT "           A4 =",A4,"        L(2) =",L(2)
1220 PRINT "         Z(1) =",Z(1),"        Z(2) =",Z(2)
1225 PRINT "         Q(2) =",Q(2),"           B =",B
1230 PRINT "           N9 =",N9
1235 REM X1 = ANFANG
1240 REM X2 = ANZAHL DER PERIODEN
1245 READ X1,X2
1250 DATA 0,50
1255 LET X2=X1+X2
1260 LET X3=1
1265 PRINT " ","    AUTONOME","   INDUZIERTE","    GESAMT-"," "," ",
1270 PRINT "TATSAECHLICHER","ERFORDERLICHER"
1275 PRINT "     PERIODE","  INVESTITIONEN","  INVESTITIONEN","  INVESTITIONEN",
1280 PRINT "       KONSUM","VOLKSEINKOMMEN","KAPITALBESTAND","KAPITALBESTAND"
1285 REM BESTIMMEN DES MINIMALEN UND MAXIMALEN Y-WERTES
1290 LET I9=0
1295 FOR X=X1 TO X2 STEP X3
1300  GOSUB  1695
1305  LET I9=I9+1
1310  IF I9 > 1 GOTO  1330
1315  LET Y1=Y
1320  LET Y2=Y
1325  LET Y4=Y
1330  IF Y > Y1 GOTO  1345
1335  IF Y < Y2 GOTO  1355
1340  GOTO  1360
1345  LET Y1=Y
1350  GOTO  1360
1355  LET Y2=Y
1360  PRINT X,K(T),J(T),I(T),C(T),Z(T),L(T),Q(T)
1365 ! ********** ALTE VERSION ********* FSOBIC VERSION 1 :*******
1370 ! PUT #DATEI#;1, X
1375 ! PUT #DATEI#;2, K(T)
1380 ! PUT #DATEI#;3, J(T)
1385 ! PUT #DATEI#;4, I(T)
1390 ! PUT #DATEI#;5, C(T)
1395 ! PUT #DATEI#;6, Z(T)
1400 ! PUT #DATEI#;7, L(T)
1405 ! PUT #DATEI#;8, Q(T)
1410 ! ********** ALTE VERSION ENDE
1415  WRITE #8,RECORD=I9: X, K(T), J(T), I(T), C(T), Z(T), L(T), Q(T)
1420 NEXT X
1425 IF N9 <= 0 GOTO  1925
1430 IF N9 > 7 GOTO  1925
1435 COMMENT FUER N9 < 1 UND N9 > 7 WIRD NICHT GEZEICHNET
1440 ! ********** ALTE VERSION ********* FOSBIC VERSION 1 *****
1445 ! RESET #DATEI#
1450 RESET #8:BEGIN
1455 PAGE
```

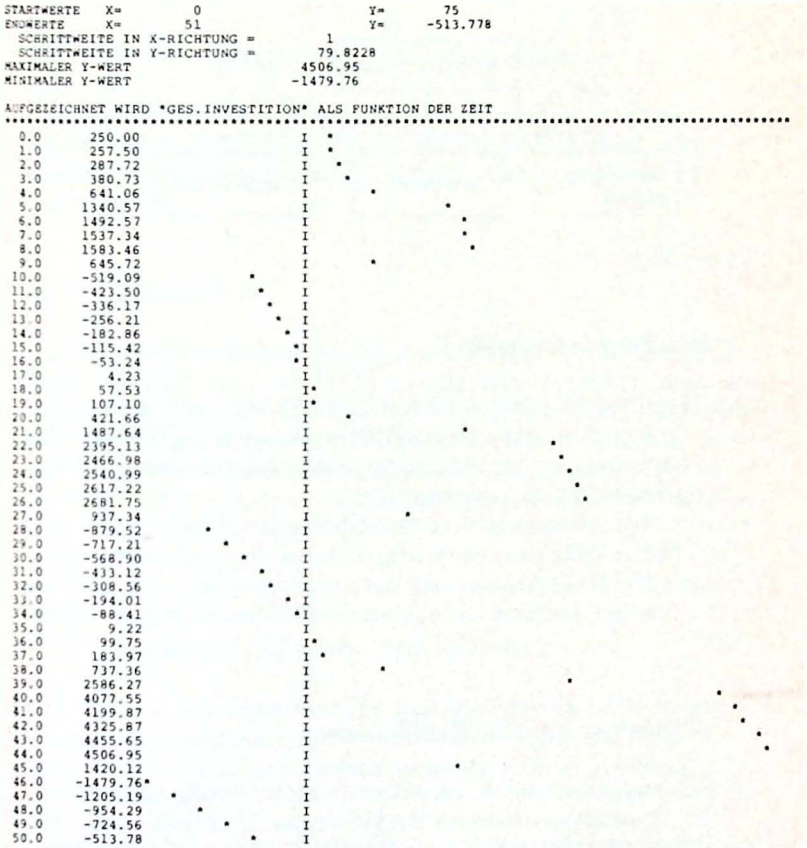
```
1460 COMMENT Y4 = MAXIMALE ANZAHL DER ZEICHENPUNKTE IN Y-RICHTUNG
1465 LET Y4=75
1470 LET Y3=ABS((Y1-Y2)/Y4)
1475 PRINT "STARTWERTE   X=",X1,"              Y=",Y4
1480 PRINT "ENDWERTE     X=",X,"              Y=",Y
1485 PRINT "  SCHRITTWEITE ","IN X-RICHTUNG =",X3
1490 PRINT "  SCHRITTWEITE ","IN Y-RICHTUNG =",Y3
1495 PRINT "MAXIMALER Y-WERT",Y1
1500 PRINT "MINIMALER Y-WERT",Y2
1505 PRINT
1510 PRINT "AUFGEZEICHNET WIRD ";M$;" ALS FUNKTION DER ZEIT"
1515 FOR I=1 TO Y4+20
1520  PRINT USING  1665  ;V$;
1525 NEXT I
1530 PRINT USING  1665  ;V$
1535 REM NULLPUNKT WENN IN ZEICHENEBENE BERECHNEN
1540 LET T1=0
1545 I9=0
1550 IF Y2 >= 0 GOTO  1565
1555 IF Y1 < 0 GOTO  1565
1560 LET T3=INT(ABS(Y2)/Y3+0.5)
1565 FOR X=X1 TO X2 STEP X3
1570 ! ********* ALTE VERSION *********** FOSBIC VERSION 1 ****
1575 ! GET #DATEI#;N9+1, Y
1580  I9=I9+1
1585  READ #8,RECORD=I9, ELEMENT=N9+1: Y
1590  LET T1=INT(AE  .Y-Y2)/Y3))+1
1595  IF T1 < T3 GOTO  1610
1600  LET T2=T1
1605  GOTO  1615
1610  LET T2=T3
1615  PRINT USING  1620  ;X;Y;
1620 %###.#  #######.##
1625  FOR I=1 TO 2
1630   IF I <>  _ GOTO  1645
1635   PRINT USING  1665  ;Z$;
1640   GOTO  1670
1645   IF I <> T1 GOTO  1660
1650   PRINT USING  1665  ;V$;
1655   GOTO  1670
1660   PRINT USING  1665  ;U$;
1665 %#%
1670  NEXT I
1675  PRINT USING  1665  ;U$
1680 NEXT X
1685 CLOSE #8
1690 STOP
1695 REM UNTERPROGRAMM DES OSZILLATIONSMODELLS NACH HICKS
1700 REM K= AUTONOME INVESTITIONEN
1705 LET T=3
1710 LET K(T)=A0*(1+R)**X
1715 REM J = INDUZIERTE INVESTITIONEN
1720 LET J(T)=V1*(Q(T-1)-L(T-1))! GEAENDERT AUF KAPITALANPASSUNG
1725 IF J(T) < A3*(1+R3)**X GOTO  1735  ! OBERGRENZE DER INDUZIERTEN INVEST.
1730 LET J(T)=A3*(1+R3)**X
1735 IF J(T) > A4*L(T-1) GOTO  1745  ! UNTERGRENZE DER INDUZIERTEN INVEST
1740 LET J(T)=A4*L(T-1)
1745 REM I = GESAMTINVESTITIONEN
1750 LET I(T)=K(T)+J(T)
1755 REM C = KOMSUM
1760 LET C(T)=C1*Z(T-1)
1765 IF C(T) <= A1*(1+R1)**X GOTO  1780  ! OBERGRENZE DES KONSUM
1770 LET C(T)=A1*(1+R1)**X
1775 GOTO  1790
1780 IF C(T) > A2*(1+R2)**X GOTO  1790  ! UNTERGRENZE DES KONSUM
1785 LET C(T)=A2*(1+R2)**X
1790 REM Z = VOLKSEINKOMMEN
1795 LET Z(T)=I(T)+C(T)
1800 REM L = TATSAECHLICHER KAPITALBESTAND
1805 LET L(T)=L(T-1)+J(T)
1810 REM Q = ERFORDERLICHER KAPITALBESTAND
1815 LET Q(T)=B*Z(T)
1820 LET Q(T-1)=Q(T)
1825 LET L(T-1)=L(T)
1830 LET Z(T-2)=Z(T-1)
1835 LET Z(T-1)=Z(T)
```

```
1840 IF N9 <= 0 GOTO  1920
1845 IF N9 > 7 GOTO  1920
1850 GOTO  1855, 1865, 1875, 1885, 1895, 1905, 1915  ON N9
1855 LET Y=K(T)
1860 GOTO  1920
1865 LET Y=J(T)
1870 GOTO  1920
1875 LET Y=I(T)
1880 GOTO  1920
1885 LET Y=C(T)
1890 GOTO  1920
1895 LET Y=Z(T)
1900 GOTO  1920
1905 LET Y=L(T)
1910 GOTO  1920
1915 LET Y=Q(T)
1920 RETURN
1925 END
```

C1 =	0.75	V1 =	1
A0 =	250	R =	0.03
A1 =	1000	R1 =	0.03
A2 =	500	R2 =	0.03
A3 =	1000	R3 =	0.03
A4 =	-.1	L(2) =	3000
Z(1) =	1000	Z(2) =	1000
Q(2) =	3000	B =	1
N9 =	1		

PERIODE	AUTONOME INVESTITIONEN	INDUZIERTE INVESTITIONEN	GESAMT-INVESTITIONEN	KONSUM	VOLKSEINKOMMEN	TATSAECHLICHER KAPITALBESTAND	ERFORDERLICHER KAPITALBESTAND
0	250	0	250	750	1000	3000	3000
1	257.5	0	257.5	750	1007.5	3000	3022.5
2	265.225	22.5	287.725	755.625	1043.35	3022.5	3130.05
3	273.182	107.55	380.732	782.512	1163.24	3130.05	3489.73
4	281.377	359.683	641.06	872.433	1513.49	3489.73	4540.48
5	289.819	1050.75	1340.57	1135.12	2475.69	4540.48	7427.06
6	298.513	1194.05	1492.57	1194.05	2686.62	5734.53	8059.85
7	307.468	1229.87	1537.34	1229.87	2767.22	6964.41	8301.65
8	316.693	1266.77	1583.46	1266.77	2850.23	8231.18	8550.7
9	326.193	319.522	645.716	1304.77	1950.49	8550.7	5851.47
10	335.979	-855.07	-519.091	1343.92	824.826	7695.63	2474.48
11	346.058	-769.563	-423.504	692.117	268.613	6926.07	805.838
12	356.44	-692.607	-336.166	712.88	376.714	6233.46	1130.14
13	367.133	-623.346	-256.212	734.267	478.054	5610.11	1434.16
14	378.147	-561.011	-182.864	756.295	573.431	5049.1	1720.29
15	389.492	-504.91	-115.418	778.984	663.565	4544.19	1990.7
16	401.177	-454.419	-53.2425	802.353	749.111	4089.77	2247.33
17	413.212	-408.977	4.23467	826.424	830.658	3680.8	2491.98
18	425.608	-368.08	57.5288	851.217	908.745	3312.72	2726.24
19	438.377	-331.272	107.105	876.753	983.858	2981.44	2951.57
20	451.528	-29.8701	421.658	903.056	1324.71	2951.57	3974.14
21	465.074	1022.57	1487.64	993.535	2481.17	3974.14	7443.52
22	479.026	1916.1	2395.13	1860.88	4256.01	5890.24	12768
23	493.397	1973.59	2466.98	1973.59	4440.57	7863.83	13321.7
24	508.199	2032.79	2540.99	2032.79	4573.79	9896.62	13721.4
25	523.444	2093.78	2617.22	2093.78	4711	11990.4	14133
26	539.148	2142.6	2681.75	2156.59	4838.34	14133	14515
27	555.322	382.014	937.336	2221.29	3158.62	14515	9475.87
28	571.982	-1451.5	-879.52	2287.93	1408.41	13063.5	4225.22
29	589.141	-1306.35	-717.21	1178.28	461.073	11757.2	1383.22
30	606.816	-1175.72	-568.901	1213.63	644.731	10581.4	1934.19
31	625.02	-1058.14	-433.124	1250.04	816.916	9523.3	2450.75
32	643.771	-952.33	-308.559	1287.54	978.982	8570.97	2936.95
33	663.084	-857.097	-194.013	1326.17	1132.15	7713.87	3396.46
34	682.976	-771.387	-88.4111	1365.95	1277.54	6942.49	3832.62
35	703.466	-694.249	9.21696	1406.93	1416.15	6248.24	4248.44
36	724.57	-624.824	99.7458	1449.14	1548.88	5623.41	4646.65
37	746.307	-562.341	183.965	1492.61	1676.58	5061.07	5029.74
38	768.696	-31.3369	737.359	1537.39	2274.75	5029.74	6824.25
39	791.757	1794.52	2586.27	1706.06	4292.34	6824.25	12877
40	815.509	3262.04	4077.55	3219.25	7296.8	10086.3	21890.4
41	839.975	3359.9	4199.87	3359.9	7559.77	13446.2	22679.3
42	865.174	3460.7	4325.87	3460.7	7786.57	16906.9	23359.7
43	891.129	3564.52	4455.65	3564.52	8020.16	20471.4	24060.5
44	917.863	3589.09	4506.95	3671.45	8178.4	24060.5	24535.2
45	945.399	474.718	1420.12	3781.6	5201.71	24535.2	15605.1
46	973.761	-2453.52	-1479.76	3895.04	2415.28	22081.7	7245.85
47	1002.97	-2208.17	-1205.19	2005.95	800.753	19873.5	2402.26
48	1033.06	-1987.35	-954.289	2066.13	1111.84	17886.2	3335.51
49	1064.05	-1788.62	-724.562	2128.11	1403.55	16097.5	4210.64
50	1095.98	-1609.75	-513.778	2191.95	1678.17	14487.8	5034.52

```
START-WERTE   X=         0                    Y=        75
END-WERTE     X=        51                    Y=      -513.778
  SCHRITTWEITE IN X-RICHTUNG =          1
  SCHRITTWEITE IN Y-RICHTUNG =       79.8228
MAXIMALER Y-WERT                   4506.95
MINIMALER Y-WERT                  -1479.76

AUFGEZEICHNET WIRD *GES.INVESTITION* ALS FUNKTION DER ZEIT
```



6 BASIC-Literatur

Die im folgenden besprochene BASIC-Literatur umfasst die aus Abbildung 31 ersichtlichen Kategorien.

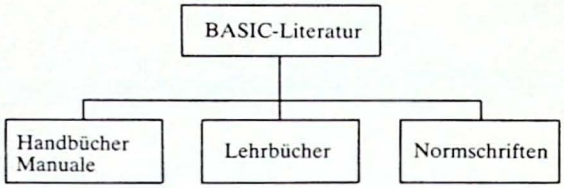


Abbildung 31.

61 Handbücher/Manuale

Trotz der Bemühungen von ANSI BASIC und ECMA um eine Standardisierung von BASIC weichen die praktisch realisierten BASIC-Sprachdialekte teilweise immer noch wesentlich voneinander ab. Dies trifft vor allem bezüglich der Sonderanweisungen zu.
Einzelheiten über systemspezifische BASIC-Versionen können den einschlägigen BASIC-Manualen der Computerhersteller entnommen werden. Entsprechende Hinweise enthält das Literaturverzeichnis, aus dem auch die für den interessierten Leser relevanten Kontaktadressen ersichtlich sind.[1]

611 Angloamerikanische Publikationen

Von amerikanischen Computerherstellern werden häufig spezielle system- resp. anwendungsorientierte BASIC-Manuale veröffentlicht. Systemspezifische Manuale stellen den auf einem Computersystem verfügbaren BASIC-Sprachumfang vollständig dar; anwendungsorientierte Manuale sind oft als leicht verständliche BASIC-Einführungen konzi-

1 Die Angaben basieren auf mehreren Umfragen; die letzte wurde im Dezember 1981/Januar 1982 durchgeführt.

piert und behandeln den Sprachumfang weniger umfassend. Eine Sonderstellung nimmt das von W. Arms und S. Hardy (1981) bearbeitete Benutzerhandbuch zu Dartmough BASIC-7 ein. Es wird an zahlreichen amerikanischen Hochschulen, die via EDUNET mit dem Kiewit Computation Center verbunden sind, benutzt und ist generell auch deshalb beachtenswert, weil die realisierte BASIC-Version – vor allem bezüglich der Matrizen- und Graphikanweisungen – bereits wesentlich über die Standardisierungsvorschläge von ANSI BASIC 82 hinausgeht.[2]

612 Kontinentaleuropäische Publikationen

Aus der kontinentaleuropäischen Literatur sind die – meist auf amerikanischen Manuales basierenden – deutschsprachigen IBM-Handbücher sowie die Sprachbeschreibung von Dietz, Kienzle und Nixdorf besonders hervorzuheben.

62 Lehrbücher

Als BASIC-Lehrbücher werden Publikationen verstanden, die sich im überwiegenden Mass mit der Programmiersprache BASIC auseinandersetzen. Ihre Zahl ist – insbesondere im angloamerikanischen Sprachbereich – sehr gross. Einen Überblick über die ältere Literatur ist aus der ersten Auflage dieser Arbeit ersichtlich.[3] Die folgenden Hinweise beschränken sich auf Neupublikationen, insbesondere aus den Jahren 1976–1982.

621 Angloamerikanische Publikationen

Unter den angloamerikanischen Lehrbüchern nimmt dasjenige von R. Conway und J. Archer (1979) eine Sonderstellung ein. Als BASIC-Einführung für Nichtprogrammierer vermittelt es im wesentlichen nur passiv

2 Vgl. EDUNET News – (1982)25, Fall, p. 1: „Special resources [available] for EDUNET members at Kiewit on DCTSL [Honeywell 66/DPS–3] include: BASIC 7, a powerful version of BASIC with special structured programming features, n-dimensional arrays, named subroutine calls, and built-in 2 and 3–D graphics routines. ... A Public Program Library featuring over 700 programs, many in BASIC, for use and/or modification by the end-user."

3 Vgl. Weber und Türschmann, BASIC 2, 1. Aufl., S. 217–226.

verwertbare Programmierkenntnisse (programming for nonprogrammers). Der Leser soll nach Durcharbeitung des Textbuches BASIC-Programme lesen und verstehen können.
In ähnlicher Weise wird auch im Lehrbuch von D. Moursund (1977) der Besprechung von Fremdprogrammen (canned programs), die zunächst in ihrer Funktionsweise zu verstehen und später den Bedürfnissen des Lesers entsprechend zu modifizieren sind, grosse Bedeutung beigemessen. Im übrigen ist das Lehrbuch speziell auf die Textverarbeitung ausgerichtet.
Normalerweise wollen BASIC-Lehrbücher aktiv verwertbare BASIC-Programmierkenntnisse vermitteln. Dabei weist die Zahl der sich ausschliesslich mit Elementaranweisungen befassenden Lehrbücher – im Vergleich zur älteren BASIC-Literatur[4] – stark rückläufige Tendenz auf. Dafür werden in zunehmendem Masse Lehrbücher veröffentlicht, die gleichzeitig in die Datenverarbeitung im allgemeinen und in BASIC einführen.
Unter den letzterwähnten EDV-Lehrbüchern sei speziell auf F. Gustavson und M. Sackson (1979) verwiesen. Das Lehrbuch stellt eine einfache, nicht mathematisch orientierte, Einführung in die computergestützte Problembearbeitung unter Verwendung von BASIC dar. Besonders hervorzuheben ist der intensive Gebrauch von Flussdiagrammen, der dem Leser den Zugang zu den einzelnen Beispielprogrammen stark erleichtert.
Das Textbuch von G. Struble (1980) basiert im wesentlichen auf ANSI Minimal BASIC; die – nur in begrenztem Masse mitberücksichtigten – Sonderanweisungen betreffen die BASIC-Version von PDP-10 (Digital Equipment Corporation). Das Buch enthält eine Reihe anspruchsvoller BASIC-Programme, insbesondere auch zur Matrizen- und Fileverarbeitung, wobei zahlreiche Aufgaben und Prüfungsfragen zur laufenden Kontrolle der Stofferarbeitung dienen.
Die sich inhaltich im wesentlichen auf BASIC-Elementaranweisungen beschränkten Lehrbücher enthalten zumeist keine systematischen Sprachdarstellungen; oft werden auch nur die von den einzelnen Autoren – subjektiv – aus relevant betrachteten BASIC-Elementaranweisungen behandelt. Die Stoffvermittlung erfolgt im allgemeinen anhand einfacher Kurzbeispiele.
BASIC-Einführungen einfachster Art stammen von M. Eagle (1970) und D. Monro (1978); das Textbuch von P. Chirlian (1978) ist extrem unübersichtlich gestaltet und lässt keine effiziente Nachbearbeitung des Stoffes zu.

4 Vgl. Weber und Türschmann, BASIC 2, 1. Aufl., S. 218 (Tabelle 9).

Ein leicht verständliches, weitgehend flussdiagrammgestütztes, BASIC-Lehrbuch mit zahlreichen beachtenswerten Anwendungsbeispielen aus der Betriebswirtschaftslehre stammt von R. Lott (1977). Auch I. Forkner (1977) nimmt in seiner – drucktechnisch wenig aufwendig gestalteten – BASIC-Einführung vorzugsweise auf Kurzbeispiele aus dem Gebiet der Betriebswirtschaftslehre Bezug.
G. Ledin (1978) misst bei Behandlung von Elementar-BASIC der strukturierten Programmierung grössere Bedeutung bei.
D. Simon (1978) behandelt in seinem Lehrbuch den gesamten BASIC-Sprachumfang, unter spezieller Berücksichtigung der Elementaranweisungen. Hervorzuheben sind die instruktiven Beispielprogramme, Übungsaufgaben und Testfragen.
Das Textbuch von R. Stair (1979) enthält eine Reihe guter Kurzprogramme; Matrizen- und Fileanweisungen werden nur kursorisch behandelt. Besonders zu vermerken ist die häufige Verwendung von Flussdiagrammen.
Das Lehrbuch von R. Lewis und B. Blakeley (1974) stellt ebenfalls eine elementare Einführung in BASIC dar, wobei auch Matrizen- und Fileanweisungen kurz behandelt werden. Beachtenswert sind die wiedergegebenen Dialogprotokolle für neun verschiedene Computersysteme (Burrough 700 Series, DC System 10, Hewlett-Packard 200 Series, IBM Call/360, ICL 1900 Series, etc.).
Auch das Textbuch von T. Logsdon (1979) führt anhand zahlreicher, vorwiegend einfach gehaltener und leicht verständlicher Beispiele in Elementar-BASIC ein; behandelt werden auch Matrizenanweisungen einfacher Art.
Unter den sich ausführlich mit Elementar- und Matrizenanweisungen befassenden BASIC-Textbüchern sticht das leicht verständliche – graphisch ungewöhnlich (in Schreibschrift) gestaltete und humorvoll illustrierte – Bändchen von D. Alcock (1977) besonders heraus.
S. Lawlor behandelt in seinem Lehrbuch (1979) die wichtigsten Elementaranweisungen und die Matrizenanweisungen. Die in den Text eingebauten Beispiele sind instruktiv und leicht verständlich. Entsprechendes gilt auch für das Lehrbuch von C. Sass (1980).
Das Lehrbuch von N. Wu (1975) stellt eine gut strukturierte Einführung in BASIC dar, in der neben den Elementar- auch die Matrizenanweisungen ausführlich behandelt werden. Beachtenswert sind die Kurzprogramme aus verschiedenen Gebieten der Betriebswirtschaftslehre.
J. Coan befasst sich in seinem BASIC-Band (1976) vor allem mit der Textverarbeitung. Bemerkenswert sind die zahlreichen – gut aufeinander abgestimmten – Kurzprogramme.
Unter den Elementar- und Sonderanweisungen behandelnden Lehrbüchern zeichnet sich die BASIC-Einführung von H. Deitel (1977) durch

besonders ausführliche Darstellung der Stringverarbeitung aus; weiterhin wird intensiv mit Flussdiagrammen und Entscheidungstabellen gearbeitet.
Das Lehrbuch von R. Forsyth (1978) ist eine leicht verständliche Einführung in Elementar-BASIC, mit Berücksichtigung der Fileanweisungen. Die zur Illustration benutzten Kurzprogramme sind durchwegs sehr instruktiv und betreffen die verschiedensten Wissensgebiete.
Auch das Lehrbuch von W. Amsbury (1981) befasst sich schwerpunktmässig mit den Fileanweisungen.
In Speziallehrbüchern behandeln – teilweise unter ausschliesslicher Bezugnahme auf Apple TM BASIC – auch L. Finkel und J. Brown (1981) die BASIC-Fileanweisungen.
In den übrigen BASIC-Lehrbüchern werden Elementar- und Sonderanweisungen ziemlich gleichgewichtig behandelt.
Eine leicht lesbare, mit zahlreichen instruktiven Beispielen ausgestattete BASIC-Einführung stammt von M. Boillot und W. Horn (1976); graphische Darstellungen erleichtern die Lektüre, Textaufgaben regen zur Bearbeitung anspruchsvoller Problemstellungen an.
Die systematisch gestaltete BASIC-Einführung von B. Bosworth und H. Nagel (1977) enthält vorwiegend wirtschaftswissenschaftlich orientierte Beispiele.
Der auf "Computer Science: A First Course" abgestimmte BASIC-Ergänzungsband von A. Forsythe u. a. (1976) enthält zahlreiche – durch Flussdiagramme unterstützte – Programme.
Die beiden BASIC-Bändchen von P. Gosling (1977, 1980) decken alle wesentlichen BASIC-Anweisungen ab; instruktive Beispiele zur Matrizen- und Fileverarbeitung werden alternativ mit DG-BASIC oder HP-BASIC bearbeitet.
Von P. Hartley (1976) stammt eine speziell für Ingenieure gedachte BASIC-Einführung; sie enthält zahlreiche einfache Anwendungsbeispiele.
E. Koffman und F. Friedman (1979) beziehen sich in ihrem Lehrbuch vor allem auf ANSI Minimal BASIC, Darthmouth BASIC und DE BASIC-PLUS. Sämtliche Beispielprogramme werden in allen drei Sprachvarianten dargestellt, wobei die Vorteile der Mitverwendung leistungsfähiger (nach ANSI BASIC standardisierter) Statements besonders deutlich wird. Hervorzuheben ist die ständige Mitbenutzung von Programmablaufplänen; ausserdem wird von den Autoren der strukturierten Programmierung grosse Bedeutung beigemessen.
Das Lehrbruch von P. Moulton (1979) enthält unkonventionelle Kurzprogramme, welche die Funktionsweise der behandelten BASIC-Statements gut verdeutlichen, dabei aber zu einem praktisch unvertretbaren Programmierstil anleiten.
Auch das Textbuch von C. Pegels und R. Verkler (1978) vermittelt einen

Gesamtüberblick über BASIC. Beachtenswert sind die zahlreichen – relativ kurzen, aber recht instruktiven – Beispiele aus dem Gebiet der Wirtschaftswissenschaft.
Das Lehrbuch von H. Walker (1980) bezieht sich auf EXTENDED BASIC-PLUS für PDP-11; als Einführung in die üblichen BASIC-Dialekte ist die Arbeit nicht geeignet.
Im Textbuch von P. Worland (1979) werden Elementar- und Sonderanweisungen gleichermassen behandelt; instruktive Kurzprogramme lassen die Funktionsweise der behandelten Statements gut erkennen, wobei auf ausführliche Dokumentation der Programmsteuerungsanweisungen – teilweise unter Realisation ungewöhnlicher Programmstrukturen – besonderen Wert gelegt wird.
Zu den amerikanischen Standardlehrbüchern gehören die Arbeiten von W. Sharpe (1979) sowie J. Kemeny und T. Kurtz (1980). Sie enthalten zahlreiche – auch für Wirtschaftswissenschaftler relevante – Beispiele und heben die Bedeutung eines guten Programmierstils deutlich hervor.
Grundlegend für die Entwicklung eines guten BASIC-Programmierstils ist die – unter Anführung positiver und negativer Beispiele – gut kom-

Tabelle 4. BASIC-Stilregeln nach J. Nevison

Rule 1	Space symbols so the line may be easily read.
Rule 2	End a block of program with a blank line.
Rule 3	Distinguish comment from code.
Rule 4	Use line numbers of equal length.
Rule 5	Sharpen comment, sharpen code, sharpen thought.
Rule 6	First draft on paper.
Rule 7	Title to tell.
Rule 8	Make a formal introduction.
Rule 9	Reference critical ideas.
Rule 10	Tag variables and constants for quick reference.
Rule 11	Match variable to ideas.
Rule 12	Label constants.
Rule 13	Initialize a variable near its use.
Rule 14	Make the line flow left to right.
Rule 15	Make the line easy to read aloud.
Rule 16	Code top to bottom.
Rule 17	Nest structures that work together.
Rule 18	Exit carefully.
Rule 19	Practice the program, exercice and proofread.

mentierte Arbeit von J. Nevison (1978). Die von ihm entwickelten 19 Stilregeln sollten allgemeine Anerkennung finden; sie sind im Wortlaut aus Tabelle 4 ersichtlich.

622 Kontinentaleuropäische Publikationen

Die Zahl der kontinentaleuropäischen BASIC-Lehrbücher ist noch relativ gering. Dies gilt auch für den deutschen Sprachbereich.
Von M. Rupp (1980) stammt eine besonders einfach gehaltene und für den EDV-Unterricht an kaufmännischen Schulen geeignete BASIC-Einführung.
Das Lehrbuch von W. Schneider (1979) führt den Leser gut in die BASIC-Elementaranweisungen ein; grössere Beispielprogramme betreffen betriebswirtschaftliche und technische Problemstellungen. Ein zweiter Band (1982) befasst sich mit den Sonderanweisungen.
Von E. Mägerle (1980) wurde speziell ein auf IBM-BASIC ausgerichtetes Lehrbuch veröffentlicht.
In der BASIC-Einführung von K. Weber 1979 werden Elementar- und Sonderanweisungen auf der Grundlage von ANSI BASIC behandelt, so dass die Beispielprogramme mit den meisten BASIC-Prozessoren nachvollzogen werden können.
Das Lehrbuch von V. Haase und W. Stucky (1977) legt auf exakte Sprachdefinitionen – im Anschluss an ANSI Minimal BASIC – besonderen Wert. Behandelt werden Elementar- und Sonderanweisungen; zu vermerken sind auch die Ausführungen über BASIC-Programmiersysteme. Beispielprogramme verdeutlichen die vielseitigen Einsatzmöglichkeiten von BASIC.
Weitere deutschsprachige BASIC-Lehrbücher stammen von G. Abeldt (1979), R. Busch (1983), G. Hamann (1982), P. Krizan und K.-D. Kaufmann (1982), S. Stief (1982) sowie von H. und Ch. Spitschka (1981); ausserdem liegen deutsche Übersetzungen der Lehrbücher von D. Alcock (1981) und B. Gottfried (1978) vor.
Aus der französischen Literatur ist das BASIC-Lehrbuch von P. Lebeux (1980) besonders hervorzuheben. Es weicht insofern von den übrigen Textbüchern ab, als nicht nur standardisierte BASIC-Formen behandelt, sondern auch Möglichkeiten zur BASIC-Weiterentwicklung – insbesondere bezüglich der Graphikanweisungen – diskutiert werden.
Die BASIC-Einführung von R. Lortal (1978) ist hinsichtlich der angeführten Beispiele speziell auf die numerische Mathematik ausgerichtet.
Von J. Lamoitier (1980) stammt eine Sammlung von fünfzig Beispielen aus verschiedenen Wissensgebieten (Mathematik, Statistik, Operations Research, etc.).

63 Normschriften

631 Angloamerikanische Publikationen

Die vom American National Standards Institute (ANSI) entwickelten Vorschläge zur BASIC-Standardisierung wurden in weitgehendem Masse in die vorliegende Arbeit integriert. ANSI Minimal BASIC bezieht sich auf die Elementaranweisungen, ANSI BASIC betrifft im wesentlichen die Sonderanweisungen.
Einzelheiten zur BASIC-Standardisierung – insbesondere auch bezüglich der grundlegenden Voruntersuchungen – sind aus Band 1 (Abschn. 14) ersichtlich; verwiesen sei auch auf den in Abbildung 3 dargestellten zeitlichen Verlauf der BASIC-Diversifikation und -Standardisierung.[5]

632 Kontinentaleuropäische Publikationen

Unter den kontinentaleuropäischen Publikationen kommt den BASIC-Standards der European Computer Manufactures Association (ECMA) zentrale Bedeutung zu.[6]
Die Standardisierungsvorschläge des niederländischen Normeninstituts sind nur noch historisch relevant.[7]

64 Literaturverzeichnis

641 Handbücher/Manuale

Die angeführten Firmenanschriften können in der Regel über die deutschen Kontaktadressen beschafft werden.

BASIC Timesharing, Inc., 650 North Mary Avenue, Sunnyvale, California 94086
- Model 4000 Interactive Timesharing System. User's Manual. Sunnyvale 1975

Burroughs Corporation, Burroughs Place, Detroit, Michigan 48232, Burroughs Deutschland GmbH, Frankfurter Allee 14–20, D-6236 Eschborn

5 Vgl. Bd. 1, S. 42–46.
6 Vgl. Weber, BASIC-Programmier-Richtlinien für Austauschprogramme, S. 14.
7 Vgl. Weber und Türschmann, BASIC 2, 1. Aufl., S. 227–231.

- B 5500 Time Sharing System. BASIC Reference Manual. Detroit 1968 = 1038064
- B 6700 BASIC Language Information Manual. Detroit 1971 = 5000383
- B 7000/B 6000 Series. BASIC Reference Manual. Detroit 1977 = 5001407
- B 4800/B 2500 Series. BASIC Reference Manual. Detroit 1978 = 1108826
- B 1800/B 1700 Systems. BASIC Reference Manual. Detroit 1979 = 1067535

CII-Honeywell Bull, 94 Avenue Gambetta, F-75020 Paris,
Honeywell Bull AG, Hohenstaufenring 62, D-5000 Köln 1

- GE-400 Series. BASIC Language for Time Sharing Programming System. Reference Manual. Phoenix 1968 = CD400H3.001.18.30.106A
- 10070 BASIC User's Manual. Prelininary ed., Louveciennes 1969 = C901.446A/EN
- International BASIC 1 Time-Sharing Manual. Bethesda 1971 = 7300.05
- System 700. BASIC OS/700. Waltham 1972 = 70130072715A
- Series 600/6000 Software. Time-Sharing BASIC. Wellesley Hills 1973 = BR36, Rev 1
- BASIC under SIRIS 7/SIRIS 8 Time Sharing. Operation and User's Manual. Louveciennes 1973 = 4041 EI/EN
- BASIC Language. MARK III Foreground Reference Manual. Bethesda 1973 = 3200.01 F
- BASIC Language. Reference Manual. Rockville 1974 = 7200.01
- Series 60 (Level 6) GCOS/BES2. BASIC Reference Manual. Louveciennes 1977 = 69A.2-AU44 A Rev. 0
- Series 60 (Level 61), Extended GCOS 61. BASIC User's Guide. Louveciennes 1978 = 16 A2 7544 Rev. 0
- Series 60 (Level 64) GCOS. BASIC. Louveciennes 1979 = 46 A2 CO43A Rev. 0
- Series 60 (Level 66)/6000. Time Sharing System Reference Manual. Louveciennes 1980 = 66 A2 DJ31 A Rev. 0
- Time-Sharing BASIC. Handbuch. Köln 1969 = 14.35.196D
- Time-Sharing Service MARK I. Time-Sharing BASIC Handbuch. 3. Aufl., Köln 1972 = 14.30.271D
- Time-Sharing Service MARK III. BASIC Language. Köln 1972 = 14.06.824A
- Informations-Service MARK I/MARK III. Time-Sharing BASIC Handbuch. 5. Aufl., Köln 1974 = 14.30.271D
- Questar/M O.S. BASIC-Kurzübersicht. Köln 1981 = 28 D2-5003, Rev. 0

- Questar/M O.S. BASIC-Handbuch. Köln 1982 = 28 D2-5004, Rev. 0
- Questar/M O.S. BAL-Handbuch. Köln 1982 = 28 D2-5010, Rev. 0

Cincinnati Milacron Inc., Morrow/Milgrove Road, Lebanon, Ohio 45036
- CIP/2200 BASIC Reference Manual. Lebanon 1974 = 7.000.0243 MA

Computer Automation, Inc., Naked Mini Division, 18651 Von Karman, Irvine, California 92713

Geveke Elektronik und Automation GmbH & Co., Gutenbergring 40, D-2000 Nordersted
- BASIC Reference Manual. ALPHA-16/LSI. Irvine 1973 = 96500-01DO
- OMNIX. TRANS-BASIC User Reference Manual. Irvine 1982 = 93180-90 Al

Computer Gesellschaft Konstanz mbH, Max-Stromeyer-Strasse 116, D-7750 Konstanz
- System TR 440. BASIC Sprachbeschreibung. Konstanz 1975 = 440.D1.10

Comshare, Inc., 3001 S. State Street, Ann Arbor, Michigan 48104
- NEWBASIC Reference Manual. Pittsburg 1971
- BASIC Reference Manual. Preliminary Version. Ann Arbor 1977 = 41291

Control Data Corporation, 215 Maffett Park Drive, Sunnyvale, California 94086

Control Data GmbH, Stresemannallee 30, D-6000 Frankfurt/Main 70
- 6000 Series Computer Systems. BASIC Language 2 Reference Manual. Minneapolis 1970 = 60306200, Rev. A
- 6000 Series Computer Systems. BASIC Language Reference Manual. Minneapolis 1970 = 60305000, Rev. A
- KRONOS III. BASIC Reference Manual. St. Paul 1970 = 41615000A
- 6000 Series Computer Systems. KRONOS 2.0 BASIC Reference Manual. Minneapolis 1971 = 59150800, Rev. B
- Cyber 70 Computer Systems Models 72, 73, 74/6000 Computer Systems. BASIC Language Reference Manual. Sunnyvale 1972 = 06306200-C
- BASIC Version 3. Reference Manual. Sunnyvale 1982 = 19983900, Rev. H

Data General Corporation, 4400 Computer Drive, Westboro, Massachusetts 01580

Data General Europe, 61 Rue de Corcelles, F-75008 Paris

Data General GmbH, Stuttgarter Str. 10, D-6236 Eschborn
- An Introduction to BASIC. Southboro 1973 = 093-00088-00
- Extended BASIC User's Manual. Southboro 1979 = 093-000065-09

Datapoint Corporation, 9725 Datapoint Drive, San Antonio, Texas 78284

Gier Electronics GmbH, Buttnerstrasse 21, D-3000 Hannover 1

- Programming Language for Cassette Systems. BASIC 2.2 Programm User's Guide. San Antonio 1973
- BASIC Language for DOS Systems. DOSBASIC User's Guide. Version 3.1. San Antonio 1976 = 50020
- Datapoint Diagnostic Language DDL. User's Guide. Version 1. San Antonio 1976 = 50249
- Datapoint 5500 DOS BASIC, BASIC 55 User's Guide. Version 2.1. San Antonio 1976 = 50172
- Datapoint DOS BASICPLUS, BASICPLS User's Guide. Version 2. San Antonio 1980 = 50335

Digital Equipment Corporation, 129 Parker Street, Maynard, Massachusetts 01754

Digital Equipment GmbH, Am Forsthaus Gravenbruch 9–11, D-6078 Neu-Isenburg 2

- Introduction to Programming. PDP-8 Family Computers. 2nd ed., Maynard 1970
- EduSystem 30 BASIC User's Guide. Mark-Sense Batch. Maynard 1971 = DEC-81-KJCA-D
- Programming Languages. 2nd ed., Maynard 1972
- BASIC-PLUS Language Manual for Use with RSTS-11 (PDP-11 Resource Sharing Time-Sharing System). 3rd ed., Maynard 1973 = DEC-11-ORBPA-B-D
- EduSystem Handbook. Maynard 1973
- BASIC/RT11 Language Reference Manual. Maynard 1973 = DEC-11-LBACA-B-D
- PDP11. BASIC-11. Language Reference Manual. Maynard 1976 = DEC-11-LIBBB-A-D
- Introduction to BASIC. Maynard 1978 = AA-0155A-TK
- PDP11. Software Handbook. Maynard 1980

Heinrich Dietz, Solinger Strasse 9, D-4330 Mülheim-Ruhr

- Dietz 621 Handbuch. Mülheim 1974
- Dietz 621 BASEX Beschreibung. Mülheim 1975
- Dietz 600 C-BASIC. Beschreibung. Mülheim 1977
- Dietz 621 Multi-User BASEX. Beschreibung. Mülheim 1977
- Dietz 621 BASIC Sprachelemente. Mülheim 1979

Educational Data Systems, 1682 Langley Avenue, Irvine, California 92714

Digital Data System GmbH (DIDAS), Münchner Strasse – Dreier-Haus, D-8013 Haar bei München

- Interactive Real-Time Information System IRIS. User Manual. Irvine 1977 = EDS 1017-10

General Automation, Inc., 1055 S. East Street, Anaheim, California 92805

ERA – General Automation GmbH, Heider-Hof-Weg 23, D-5100 Aachen–Verlautenheide
- BASIC-16 Reference Manual. Anaheim 1974 = 88A00256A-C
Harris Corporation, Computer Systems Division, 2101 Cypress Creek Road, Fort Lauderdale, Florida 33309
Harris GmbH, Rennbahnstrasse 71–74, D-6000 Frankfurt/Main 71
- Harris Extended BASIC. Reference Manual. Fort Lauderale 1981 = 305/974-1700, Rev. C, Change 1
Hewlett-Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, California 95014
Hewlett-Packard GmbH, Herrenberger Strasse 110, D-7030 Böblingen
- BASIC Language Reference Manual for Hewlett-Packard 2114A/B, 2115A and 2116B Computers. Palo Alto 1968
- 2000A: A Guide to Time Shared BASIC for Reference and Self-Instruction. Cupertino 1969 = HP 02000-90002
- Time-Shared BASIC Programm Library Handbook. Cupertino 1971 = 36000A, Option DOO
- Time-Shared BASIC Handbook. November 1971 Supplement. Cupertino 1971 = 36000-90002
- 2000F: A Guide to Time Shared BASIC. Cupertino 1973 = 02000-90044
- BASIC/1000L Real-Time BASIC. Reference Manual. Cupertino 1980 = 2RTEL.320.92076-90001
- Time-Sharing BASIC. Nieder Eschbach 1970
- System 35 Tischcomputer. Bedingungs- und Programmier-Handbuch. Böblingen 1980 = 09835-90101
- System 45. Programmier- und Bedienungshandbuch. Fort Colans 1980 = 09845-91101
Honewell Information Systems Inc., 5250 West Century Boulevard, Los Angeles, California 90045
- CP-6 BASIC Reference Manual. Westwood 1981 = CE32-02
International Business Machines Corporation, Old Orchard Road, Armonk, New York 10504
IBM Deutschland GmbH, Pascalstrasse 100, D-7000 Stuttgart 80
- CALL/360: BASIC Introduction. White Plains 1968 = J20-0035-1
- BASIC Language Reference Manual. New York 1970 = GC28-6837-0
- IBM System/360 OS/DOS ITF: BASIC Terminal User's Guide. 2nd ed., New York 1970 = SC28-6835-1
- IBM System/360 OS(TOS)ITF: BASIC Terminal User's Guide. New York 1971 = SC28-6840-0
- System/3 Model 6. Guide to BASIC. Learner-Paced Format. Student Text. 2nd ed., New York 1971 = SR29-5001-1

- IBM System/3 Model 6. System/3 BASIC Reference Manual. 4th ed., New York 1972 = GC34-0001-3
- CALL-OS: BASIC Language Reference Manual. 4th ed., New York 1972 = GH20-0699-3
- CALL: BASIC Language Reference Manual. 3rd ed., New York 1973 = GE19-5017-2
- Business Analysis/BASIC. General Information Manual. 3rd ed., New York 1973 = GH20-1175-2
- IBM 5100. BASIC Introduction. 2nd ed., White Plains 1975 = SA21-9216-1
- IBM System/370. VS BASIC Language. 3rd ed., New York 1976 = GC28-8303-2
- IBM 5100. BASIC Reference Manual. 4th ed., White Plains 1977 = SA21-9217-3
- IBM 5110. BASIC Introduction. White Plains 1978 = SA21-9306-0
- IBM 5110. BASIC Reference Manual. 2nd ed., Atlanta 1978 = SA21-9308-1
- IBM 5110. BASIC User's Guide. 3rd ed., White Plains 1979 = SA21-9307-2
- IBM 5110. BASIC Reference Manual. 3rd ed., White Plains 1979 = SA21-9308-2
- CALL: BASIC Handbuch. Stuttgart 1971 = C12-1030-1
- System/3 Modell 6. BASIC Programmiersystem. Lehrbuch zum Selbststudium. Stuttgart 1971 = R12-3140-0
- IBM System/3. Einführung in System/3 Modell 6 BASIC. Stuttgart 1971 = GC12-3027-0
- IBM System/3 Modell 6. System/3 Handbuch. Stuttgart 1973 = GC12-3034-2
- IBM 5100. BASIC Einführung. Stuttgart 1976 = SA12-2151-0
- IBM 5100. BASIC Benutzerhandbuch. Stuttgart 1976 = SA12-2150-1
- VSPC im IBM Servicezentrum. VS BASIC. Benutzerhandbuch. Stuttgart 1977 = GH12-1386-0
- IBM System/34. BASIC Handbuch. Stuttgart 1979 = SC12-3539-0
- Tischcomputer IBM 5110/IBM 5120. BASIC Benutzerhandbuch. Stuttgart 1980 = SA12-2270-0
- IBM System/34. BASIC Handbuch. 3. Aufl., Stuttgart 1981 = SC12-3539-2
- BASIC-Handbuch für den Informatik-Unterricht in Schulen. 3. Aufl., Bonn-Bad Godesberg 1974 = F12-1030-2

International Computers Limited, ICL House, Putney, London SW15 15W

ICL Deutschland International Computers GmbH, Marienstrasse 10, D-8500 Nürnberg 1

- 1900 Series. BASIC Reference Manual. London 1971 = 4281
- System 4. Multijob BASIC. London 1973 = 4561
- Introduction to BASIC-K. London 1978 = 5641

ITS International Timesharing Corporation, ITS Building, Jonathan Industrex, Chaska, Minnesota 55318
- ITS EXBASIC. Reference Manual. Chaska 1971

Kienzle Apparate GmbH, Prinz-Eugen-Strasse 20, D-7730 VS-Villingen
- System 6100. Programmierhandbuch BASIC 6100. Villingen 1974 = 3540-60-67-000-01/0
- BASIC TI990 Reference Manual. Villingen 1979 = 2250 304-9701

MAI Management Assistance Inc., 300 East 44th Street, New York, New York 10017
MAI International GmbH, Saonestrasse 1, D-6000 Frankfurt/Main
- MAI basic/four. Business BASIC. Programmierung. Frankfurt 1975 = 210 1 75

Micro V Corporation, 17791 Skypark Circle, Irvine, California 92714
Digital Data System GmbH (DIDAS), Münchner Strasse – Dreier-Haus, D-8013 Haar bei München
- STARDOS BASIC Manual. Irvine 1979 = P001-REV B0

Modular Computer Systems, Inc., 1650 West McNab Road, Fort Lauderdale, Florida 33309
- Reference Manual BASIC. Fort Lauderdale 1974

NCR Corporation, 1700 South Patterson Boulevard, Dayton, Ohio 45479
NCR GmbH, Ulmer Strasse 160, D-8900 Augsburg
- NCR Century BASIC-2. Student Text. Dayton 1973 = DP 9825
- NCR BASIC Language Reference Manual. Dayton 1981 = RM-0188
- NCR BASIC-I-Systeme. Augsburg 1979

Nixdorf Computer AG, Fürstenallee 7, D-4790 Paderborn
- Systemliteratur Nixdorf 8870/1. System-Software Business-Basic. Paderborn 1979 = 34001.00.4.93-01
- Systemliteratur Nixdorf 8870. Structured BASIC. Paderborn 1981 = 34213.00.9.93

Ing. C. Olivetti & Co., S.p.A., Via G. Jervis 77, I-10015 Ivrea
Deutsche Olivetti GmbH, Lyoner Strasse 34, D-6000 Frankfurt/Main 71
- P 6060 Reference Manual. Preliminary edition, Ivrea 1976
- P 6066 Reference Manual. Ivrea 1980 = 3946870 Z (0)
- BCS 2000. BASIC Language Reference Manual. Ivrea 1981 = 3976990 G(2)
- BCS 2000. Business Computer Operating System (BCOS). BASIC Environment. Programmer Manual. Ivrea 1981 = 3977070 G(2)
- P 6060 Anwenderhandbuch. [Frankfurt] 1976
- P 6060/P 6066 Allgemeines Handbuch. Frankfurt 1980 = 04/521/P/3.81
- P 6060/6066 Lehrbriefe BASIC. Frankfurt 1980 = 04/505/P/5.80

Perkin-Elmer, Computer Operations, 2 Crescent Place, Oceanport, New Jersey 07757
Perkin-Elmer Data Systems GmbH, Seidlstrasse 8, D-8000 München 2
– BASIC Level II. Reference Manual. Oceanport 1978 = 29-488R04
Philips Data Systems B.V., O. Apeldoornse Weg, NL-7300 AE Apeldoorn
– P/9200 Time Sharing BASIC. Apeldoorn 1973 = 5122 911 23071
– Module KO8. BASIC Programmer's Guide. User Library. Apeldoorn 1976 = 5122 993 31831
– P 800M Stand Alone BASIC Manual. Apeldoorn 1976 = 5122 991 25482
– P 800M BASIC Manual. Apeldoorn 1978 = 5122 991 27991
– P/9200 Time Sharing System. BASIC. Düsseldorf 1971 = 5112 900 30011
– BASIC 350-K Compiler for P350-S. [Siegen] 1974 = 8617 210 36713
Prime Computer, Inc., Prime Park, Natick, Massachusetts 01760
Prime Computer GmbH, Gustav-Stresemann-Ring 12–16, D-6200 Wiesbaden
– BASIC Interpretive Language. Natick 1974
– IDR 1813. The Interpretive BASIC Programmer's Guide. Framingham 1978 = P/N MAN3251-001
Siemens Aktiengesellschaft, Hofmannstrasse 51, D-8000 München 70
– Siemens-System 7.700, Siemens-System 4004. Betriebssystem BS 2000. Dialog-BASIC. Beschreibung. 3. Aufl. (Version 5.0), München 1979 = D15/5313-03
Société Logabax, 79 Avenue Aristide Briand, F-94110 Arceuil
– LX 500. BDOS BASIC-N. Manuel de Référence. Arceuil 1980 = MR-F-00510
Sperry Corporation, Sperry Univac Division, Sperry Building, 1290 Avenue of the Americas, New York, New York 10104
Sperry GmbH, Geschäftsbereich Sperry Univac, Finkenweg, D-6231 Sulzbach
– Sperry UNIVAC 1100 Series. BASIC Programmer Reference. St. Paul 1978 = UP-7925, Rev. 2
Sperry Univac Minicomputer Operations (formerly Varian Data Machines), 2722 Michelson Dr., Irvine, California 92664
– Varian Software Handbook. Irvine 1971
Wang Laboratories, Inc., One Industrial Avenue, Lowell, Massachusetts 01851
Wang Laboratories GmbH, Moselstrasse 4, D-6000 Frankfurt/Main
– 2200 B BASIC Language Specification. Tewksbury 1973
– Wang BASIC Language Reference Manual. Lowell 1976 = 700-3038H

Westinghouse Electronic Corporation, Digital Products Department, 1200 West Colonial Drive, Orlando, Florida 32804
Westinghouse Electric SA, Computer and Instrumentation Division, Burymead Road, Hitchin, Hertfordshire GS5 1RT
- BASIC II & III Reference Manual. Orlando 1973 = 25REF-012A
Xerox Data Systems. Vgl. Honeywell Information Systems Inc.
- Xerox BASIC Sigma 5-9 Computers. Language and Operations Reference Manual. El Segundo 1971 = 90 15 46D

642 Lehrbücher

Abeldt, Günter. BASIC. Grundlagen und Beispiele. Anwendung in Mikro- und Minicomputern. Stuttgart: Frech, 1979
Ackermann, Hanns. BASIC in der medizinischen Statistik. Braunschweig: Vieweg, 1977
Alcock, Donald. Illustrating BASIC (A Simple Programming Language). Cambridge: Cambridge University Press, 1977
Alcock, Donald. Das kleine BASIC Handbuch. Paderborn: Schöningh, 1981
Amsbury, Wayne. Structured BASIC and Beyond. London: Pitman, 1981
Boillot, Michel and Horn, Lister W. BASIC. St. Paul: West Publishing Co., 1976
Bosworth, Bruce and Nagel, Harry L. Programming in BASIC for Business. Chicago: Science Research Associates, 1977
Brauch, Wolfgang. Programmierung mit BASIC. Stuttgart: Teubner, 1981
Busch, Rudolf. BASIC für Einsteiger. München: Franzis, 1983
Chirlian, Paul M. Beginning BASIC. Portland: Dilithium Press, 1978
Coan, James S. Advanced BASIC. Applications and Problems. Rochelle Park: Hayden Book, 1976
Conway, Richard and Archer, James. Programming for Poets. A Gentle Introduction Using BASIC. Cambridge: Winthrop Publishers 1979
Dederichs, Wolfgang. APPLESOFT-BASIC. Mannheim: Bibliographisches Institut, 1982
Deitel, Harvey M. Introduction to Computer Programming with the BASIC Language. Englewood Cliffs: Prentice-Hall, 1977
DeRossi, Claude. Learning BASIC Fast. 2nd ed., Reston: Reston Publishing Co., 1979
Dock, V. Thomas. Structured BASIC. Programming for Business. 2nd ed., St. Paul: West Publishing Co., 1982

Dwyer, Thomas A. and Chritchfield, M. A Bit of BASIC. Reading: Addison-Wesley, 1980
Eagle, M. R. An Introduction to BASIC. London: Bell, 1976
Fegan, Richard C. and Brosche, Susan L. BASIC Computer Programming for Statistics. A Manual. Englewood Cliffs: Prentice-Hall, 1979
Feichtinger, Herwig. BASIC für Mikrocomputer. München: Franzis, 1982
Finkel, LeRoy and Brown, Jerald R. Data File Programming in BASIC. New York: Wiley, 1981
Finkel, LeRoy and Brown, Jerald R. Apple TM BASIC. Data File Programming. New York: Wiley, 1981
Forkner, Irvine H. BASIC Programming for Business. Englewood Cliffs: Prentice-Hall, 1977
Forsyth, Richard. The BASIC Idea. An Introduction to Computer Programming. London: Chapman and Hall, 1978
Forsythe, A. I.; Hughes, C. E.; Aiken, R. M. and Organick, E. I. Computer Science. Programming in BASIC. New York: Wiley, 1976
Gala, Basil E. The Computation of Probability with BASIC. San Francisco: Holden-Day, 1977
Gosling, Peter E. Beginning BASIC. London: Macmillan, 1977
Gosling, Peter E. Continuing BASIC. London: Macmillan, 1980
Gosling, Peter E. Program Your Microcomputer in BASIC. London: Macmillan, 1981
Gottfried, Bryon S. Programming with BASIC. New York: McGraw-Hill, 1975
Gottfried, Bryon S. Programmieren mit BASIC. Theorie und Anwendung. Düsseldorf: McGraw-Hill, 1978
Graham, Neill. Computers and Computing. An Introduction Through BASIC. St. Paul: West Publishing Co., 1982
Groeneveld, Richard A. An Introduction to Probability and Statistics Using BASIC. New York: Dekker, 1979
Gustavson, Frances G. and Sackson, Marian V. Problem Solving and BASIC. A Modular Approach. Chicago: Science Research Associates, 1979
Haase, Volkmar und Stucky, Wolffried. BASIC. Programmieren für Anfänger. Mannheim: Bibliographisches Institut, 1977
Hamann, Günter O. Datenverarbeitung mit BASIC. 2. Aufl., Gernsbach: Deutscher Betriebswirte-Verlag, 1982
Hartley, P. J. Introduction to BASIC. A Case Study Approach. London: Macmillan, 1976
Hubin, Wilbert N. BASIC Programming for Scientists and Engineers. Englewood Cliffs: Prentice-Hall, 1978

Hull, Thomas E. and Day, D. D. An Introduction to Programming and Applications with BASIC. Don Mills: Addison-Wesley, 1978
Humphrey, Thomas J. BASIC Programming for the Financial Executive. New York: Wiley, 1978
Kemeny, John G. and Kurtz, Thomas E. BASIC Programming. 3rd ed., New York: Wiley, 1980
Koffman, Elliot B. and Friedman, Frank L. Problem Solving and Structured Programming in BASIC. Reading: Addison-Wesley, 1979
Krizan, Peter und Kaufmann, Klaus-Dieter. Spass mit BASIC für Anwender. Puchheim: Idea, 1982
Lamoitier, Jean-Pierre. Le basic par la pratique (50 exercises). Paris: Sybex, 1980
Lawlor, Steven, C. BASIC. Belmont: Wadsworth, 1979
Lebeux, Pierre. Introduction au BASIC sur micro-ordinateurs. Paris: Sybex, 1980
Ledin, George, Jr. A Structured Approach to General BASIC. San Francisco: Boyd & Fraser, 1978
Lewis, R. and Blakeley, B. H. Elements of BASIC. 2nd ed., Manchester: NCC Publications, 1974
Lortan, Robert. Programmation conversationnelle BASIC. Analyse numérique. Paris: Masson, 1978
Logsdon, Tom. Programming in BASIC. With Applications. 3rd print., Fullerton: Anaheim Publishing Co., 1979
Lott, Richard W. BASIC with Business Applications. New York: Wiley, 1977
Mägerle, Erich W. Einführung in das Programmieren in BASIC. 2. Aufl., Berlin: de Gruyter, 1980
Marateck, Samuel L. BASIC. 2nd ed., New York: Academic Press, 1982
Menzel, Klaus BASIC in 100 Beispielen. Stuttgart: Teubner, 1982
Monro, Donald M. Basic BASIC. London: Arnold, 1978
Morton, Jeffrey, B. Introduction to BASIC. London: Pitman, 1979
Moulton, Peter. Foundations of Programming Through BASIC. New York: Wiley, 1979
Moursund, David. BASIC Programming for Computer Literacy. New York: McGraw-Hill, 1978
Nevison, John M. The Little Book of BASIC Style. Now to Write a Program You Can Read. Reading: Addison-Wesley, 1978
Osborne, Adam; Eubranks, Gorden, Jr. and McNiff, Martin. BASIC User Guide. Berkeley: Osborn/McGraw-Hill, 1981
Pegels, C. Carl and Verkler, Robert C. BASIC. A Computer Programming Language with Business and Management Applications. 3rd ed., San Francisco: Holden-Day, 1978

Poole, Lon and Borchers, Mary. Some Common BASIC Programs. Berkeley: Osborne, 1977
Rupp, Martin. BASIC. Eine Einführung in die Programmierung mit Übungsbeispielen und Aufgaben für den kaufmännischen Unterricht. 3. Aufl., Eggenstein-Leopoldshafen: Mende, 1980
Sass, C. Joseph. BASIC Programming and Applications. Boston: Allyn & Bacon, 1980
Schneider, Wolfgang. Einführung in BASIC. 2. Aufl. Braunschweig: Vieweg, 1982
Schneider, Wolfgang. BASIC für Fortgeschrittene. Braunschweig: Vieweg, 1982
Sharpe, William F. BASIC. An Introduction to Computer Programming Using the BASIC Language. 3rd ed., New York: Macmillan, 1979
Simon, David E. BASIC from the Ground Up. Rochelle Park: Hayden, 1978
Spitschka, H. und Spitschka, Ch. BASIC mit 29 Dateiprogrammen. Ein Programmierkurs für die betriebliche Praxis. München: CW-Publikationen, 1981
Stair, Ralph M., Jr. Programming in BASIC with Cases and Applications for Profit and Nonprofit Organizations. Homewood: Irwin, 1979
Stief, Siegfried. BASIC. Systematische Darstellung für den Anwender. 2. Aufl., München: Oldenbourg, 1982
Struble, George. Business Information Processing with BASIC. Reading: Addison-Wesley, 1980
Walker, Henry M. Problems for Computer Solutions Using BASIC. Cambridge: Winthrop, 1980
Weber, Karl. BASIC für Anfänger. Bern: Haupt, 1979
Wissebach, Bertold. Entscheidungen der Produktionsplanung mit einer Sammlung von BASIC-Programmen. Basel: Birkhäuser, 1978
Wohl, Gerald. BASIC Language Programming for Everyone. New York: Pace & Pace, 1975
Worland, Peter B. Introduction to BASIC Programming. A Structured Approach. Boston: Houghton Mifflin, 1979
Worth, Thomas. BASIC for Everyone. Englewood Cliffs: Prentice-Hall, 1976
Wu, Neas L'Abbe. BASIC. The Time-Sharing Language. Dubuque: Brown, 1975

643 Normschriften und übrige Basismaterialien

ANSI, American National Standards Institute. American National Standard for Minimal BASIC. New York: ANSI, 1978 = ANSI X3.60-1978

ANSI, American National Standards Institute. American National Standard for BASIC. Preliminary Draft. September 1979. Hanover 1979 = X3J2/79-51

ANSI, American National Standards Institute. American National Standard for BASIC. Draft Proposed. February 15, 1982. Hanover 1982 = X3J2/82-06

ANSI, American National Standards Instituts. Comments and Responses (Preliminary draft). June 15, 1982. Hanover 1982 = X3J2/82-12

Arms, William Y. and Hardy, Susan E. BASIC 7 User's Guide. Preliminary edition. Hanover: Dartmouth College Computing, Kiewit Computation Center, 1981 = TM 115

Bull, G. M.; Freeman, W. and Garland, S. J. Specification for Standard BASIC. Manchester: National Computing Center, 1973 (out of print)

ECMA, European Computer Manufacturers Association. Standard ECMA-55 Minimal BASIC. Geneva: ECMA, 1978

Garland, Stephen J. Dartmouth BASIC: A Specification. Hanover: Dartmouth College, 1973 = TMO28

Henshon, Elaine M. VIEWIT: A Take-Apart Compiler. M.S. Thesis. Worcester: Worcester Polytechnic Institute, 1973

Isaacs, Gerald L. Interdialect Translatabiblity of the BASIC Programming Language. Iowa City: American College Testing Program, 1972 = ACT Technical Bulletin No. 11

Isaacs, Gerald L. BASIC Revisited. An Update to Interdialect Translatability of the BASIC Programming Language. Iowa City: CONDUIT, 1976

ISO, International Organization for Standardization. Second Draft Proposal ISO/DP 6373. Minimal BASIC. Geneva: ISO, 1979

Kemeny, John G. and Kurtz, Thomas E. BASIC. User's Manual. [1st ed.,] Hanover: Dartmouth College Computation Center, 1964

Kemeny, John G. and Kurtz, Thomas E. BASIC. 5th ed., Hanover: Kiewit Computation Center, Dartmouth College, 1970

Luehrmann, Arthur. An Elaboration of Some Thoughts on a Graphical Syntax in BASIC. Hanover: Dartmouth College, 1974

Nederlands Normalisatie-instituut. Automatische gegevensverwerking. Programmeertaal Elementair Basic. Rijswijk: Nederlands Normalisatie-instituut, 1975 = NPR 3591

Nederlands Normalisatie-instituut. Proposal for ISO-Standard on Programming Language Elementary BASIC. Rijswijk: Nederlands Normalisatie-instituut, 1976 = ISO/TC 97/SC5 (Netherlands)

Ohlmacher, George; Cassels, Robert; Hall, Thomas; Stanley, Brian and Beam, Alfred. BASIC for the UNIVAC 1108. College Park: University of Maryland, Computer Science Center, 1968 = Technical Report 68-77, NsG-398

Sharpe, William F. University of Washington BASIC Interpretive Compiler UWBIC. Seattle: University of Washington, Graduate School of Business Administration, 1967 = Technical Reports Series, No. 3

Weber, Karl. BASIC Programmier-Richtlinien für Austauschprogramme. Bern: Haupt, 1979

Weber, Karl und Türschmann, Carl Wolfram. FOSBIC-Compiler. Ein BASIC-Compilersystem auf FORTRAN-Basis. Bern: Haupt, 1977

7 Stichwortverzeichnis

Eine alleinstehende Zahl verweist immer auf eine Seite des entsprechenden BASIC-Bandes. Bei einer durch einen Punkt (.) unterteilten Zahlengruppe verweist die erste Zahl auf die Seite und die zweite Zahl auf die Definitionsgleichung. Sind Angaben durch einen oder mehrere Schrägstriche getrennt, so verweisen die einzelnen Angaben auf verschiedene Bereiche. Ein Bindestrich (–) zwischen Zahlen zeigt das entsprechende Seitenintervall an.

	BASIC 1	BASIC 2	BASIC 3
Abfrageanweisung, Graphische			119.357
ABS (Absolutwert)	108		
ACOS (Arcuscosinus)	111		
ACOT (Arcuscotangens)	112		
ARCOS (Areacosinus)	112		
ARCOT (Areacotangens)	112		
ARSIN (Areasinus)	112		
ARTAN (Areatangens)	112		
ASIN (Arcussinus)	111		
ATN (Arcustangens)	108		
ATN2	111		
Adresse		136.275	63.275
Aktualparameter		171.301	
- variable		171.303	
- wert		171.302	
Aktualparameterliste,		170.300	127
- Alphanumerische	114.66		32
- Hybrid alphanumerische	117.68g		32
- Numerische	113.63		32
- Hybrid numerische	119.68k		32
Alphazeichen	68.2		
Alternativbetrieb	32		178
Anweisung	122.69		
- Ausfuehrbare	129.78		
- Nicht ausfuehrbare	130		
Arbeitsmodus (ACCESS)		129.268	58
ASK BACKGROUND COLOR-Statement			103
ASK BEAM-Statement			120.359
ASK BOUNDS-Statement			103
ASK CHARACTER HEIGHT-Statement			121.361
ASK CLIP-Statement			104
ASK COLOR-Statement			96
ASK COLOR COUNT-Statement			96
ASK DEVICE BOUNDS-Statement			103
ASK FL-Statement		161.290	80.290
ASK JUSTIFY-Statement			121.360
ASK LINE STYLE-Statement			103
ASK POSITION-Statement			120.358
ASK STYLE COUNT-Statement			103
ASK TEXT ANGLE-Statement			122.362
ASK VIEWPORT-Statement			97
ASK WINDOW-Statement			97
Attribut		203.293	80
Ausgabeanweisung,	173.122		41/167.412
			169.416
- Graphische			106.341
Ausgabeliste	174.124		117
- Allgemeine	195.151		
- Abschluss-	179.125		
- TAB-	183.127		
Ausdruck,	94.48		
- Alphanumerischer	101.50		
- Arithmetischer	96.49		
- komparativer	104.54		
- Logischer	102.53		
- Logisch-komparativer	102.52		
Ausfuehrungsueberwachung			63
Ausnahmeblock			47
Auswahlblock			45

	BASIC 1	BASIC 2	BASIC 3
BASE-Statement	141.99		
BASIC-			
- Anwendungsbeispiele	246-274		190-255
- Alternativbetrieb	32		
- Charakteristik	14	12	12
- Dialogbetrieb	14/62		
- Hochschuleinsatz		25	
- Initialisierungsanweisung			163.407
- Literatur			256-276
- Minicomputer		13	
- Monologbetrieb	27-30		
- Problembearbeitung	34	47	14
- Programm	244.199		
- Programmiersystem			12
- Sprachanweisung		12.204	
- Sprachausschoepfung	63		
- Sprachstruktur	65	45	27
- Sprachsystem		12	
- Sprachumfang		25	
- Standardisierung	42	30	
- Struktur	65	45	27
- Systemanweisung			158
Basis	141.99		
Benutzerprogramm		28	
Betriebswirtschaftslehre	246-254		211-240
Bibliotheksname			170.436
Bibiliothekverwaltungsanweisung			177.432
Bildpunkt-			86/93/104
- abfrageanweisung			102.337
- angabeanweisung			99.333
- anweisung			98.332
Bildunterprogrammanweisung			126.366
BNF-Zeichen	65		
BREAK-Statement			141.382
Buendigkeitsregler	193.146		
Buchstabe	68.3		28
Call PLOT-Statement			127.367
CALL-Statement		169.297	83
CASE-Statement			46
CASE ELSE-Satement			46
CATALOG-Statment			184.446
CATLIST-Statement			185.447
CATPRINT-Statement			185.448
CEIL (Ganzzahl)	111		
CHAIN-Statement		182.316	
CHANGE-Statement			173.427
CHA$ (Zeichenersetzung)	115		
Chi-Quadrat-Test			195-197
CHR$	120		
CLEAR-Statement			101.336
			176.431
CLOSE FL-Statement		132.271	58
Codierblaetter	128		
Codierung	58		
COLLATE-Statement			38
COMMENT-Statement	132.84		
COMPILE-Statement			164
CONTINUE-Statement			48
COS (Cosinus)	107		
COSH (Hyberbolischer Cosinus)	111		
COT (Cotangens)	111		
COTH (Hyperbolischer Cotangens)	112		
Critical Path Method (CPM)	250-254		
Cross-Reference-Liste			186-189
CSC (Cosekans)	111		
Data-Kostante	159.112		
DATA-Statement	159.110		
data-Statement	162.114		
DATE (Datum, numerisch)	111		
DATE$ (DATUM, alphanumerisch)	115		
Datenflussplan	48		
DEBUG OFF-Statement			137.378
DEBUG ON-Statement			136.377
DEF-Block	147-157		35
Definition	66		

	BASIC 1	BASIC 2	BASIC 3
DEF-Statement,	147.100		35
- Alphanumerisches	152.105		
	153.108		
- Hybrides	154.108a		
- Hybrid alphanumerisches	155.108c		
	157.108i		
- Hybrid numerisches	154.108b		
	156.108h		
- Numerisches	147.101		
	151.104		
DEG (Bogenmassumrechnung)	111		
Deklarationen,	139.96		37
- Alphanumerische	137.92		
- Numerische	135.87		
DECLARE-Statement			39
DELETE FL-Statement			71-72
DELETE-Statement			71/174.429
DEL$ (Zeichenloeschung)	115		
Determinante		98	
Diagnoseanweisung		12	135.375
Dialogbetrieb	14/59/61		161
DIM-Statement,	134.86		33
- Allgemeines	139.95		
- Alphanumerisches	137.94		
- Numerisches	136.91		
DIM-Stringlaenge		65.224b	
DIM-Wert		61.222	
Dimensionierung	135.88		
DISPLAY (zeichenstrukturiert)		119	55
Druck-Ausgabesteuerung	147-149		
Druckzone	144-145		
DUMP-Statement			143.384
EDIT-Statement			163
Einermatrix		94	
Eingabeanweisung,	158.109		39
- Graphische			122.363
Einheitsmatrix		95	
Eintragung,		58	
- Alphanumerische		58.218	
- Numerische		58.217	
Elementaranweisung	129	47.206	28
Elementarfunktionen	107.58		
Elementnummer		136.278	
Eliminationsanweisung			182.442
ELSE-Statement	223.175a		
END DEF-Statement	151/153/157		36
END FUNCTION-Statement			84
END IF-Statement	223.175a		
END LOOP-Statement	233.184		44
END LOOP UNTIL-Statement			44
END LOOP WHILE-Statement			44
END-Statement	235.188		
END WHEN-Statement			48
EPS	111		
ERASE FL-Statement			60
EXIT DEF-Statement			36
EXIT FOR-Statement			43
EXIT FUNCTION-Statement			84
EXIT HANDLER-Statement			48
EXIT PICTURE-Statement			132.373b
EXIT-Statement	234.185		44
EXIT SUB-Statement		177.313	83.313
EXP (Exponent zur Basis von e)	108		
Exponententeil	80.29		29
Exponentielle Glaettung	246-249		
EX-Zeichen	188.142		
EX-Zeichenkette	188.141		
E-Zeichenkette	188.139		
Feldname	86.40/87.41		
File-			
- anweisung		116-165	53-82
- attribut		127.264	58.264
- ausgabeanweisung		135.273	61.273
- eingabeanweisung		148.283	72.283
- kontrollanweisung		156.288	78.288
- modifikationsanweisung			68-72

	BASIC 1	BASIC 2	BASIC 3
- name		126.262	59.262
- organisation (ORGANIZATION),		121-123	56-58
		128.267	59.267
-- Fortlaufende (STREAM)			57
-- Indizierte (KEYED)			57
-- Relative (RELATIVE)		123	57
-- Sequentielle (SEQUENTIAL)		122	57
- parameterliste		127.263	
- struktur (RECTYPE),		119-121	55-56
-- Feldstrukturierte (TEMPLATE)			55
-- Inhaltstrukturierte (INTERNAL)		120-121	55
-- Zeichenstrukturierte (DISPLAY)		119-120	55
- typ		128.266	59.266
- vereinbarung		124.259	58.259
FLMAX (Groesster Maschinenexponent)	112		
Formalparameterliste,		174.309	132
- Alphanumerische	153.107		
- Hybrid alphanumerische	155.108e		
- Hybrid numerische	156.108g		
- Numerische	148.103		
Formalparameter		174.310	
FOR-Statement	224.176		43
FOR/NEXT-Schleife	224-225/229		
	52-53		
- Verschachtelte	229-230		
FOR/NEXT-Statement	52/224.176		43-44
FOSBIC	30-31/38-42		
FOSBIC-Compilersystem	38-42	29	158-186
FP	111		
FRA (Dezimalteil)	112		
Funktion,	106-121		
- Alphanumerische	114.64		
- Benutzerdefinierte alphanumerische	115.68		
- Benutzerdefinierte numerische	113.62		
- Eingebaute alphanumerische	114.65		
- Eingebaute numerische	107.57		
- Numerische	107.56		
FUNCTION-Block			83
FUNCTION-Statement			84
Funktionsidentifikator,			
- Alphanumerischer	153.106		
- Hybrid alphanumerischer	156.108f		
- Hybrid numerischer	155.108d		
- Numerischer	148.102		
Funktionsname	113.62a		
F-Zeichenkette	188.138		
Ganzzahl,	76/77.21		
- E-Formatierte	77.25		28.25
- I-Formatierte	77.22		
- Vorzeichenlose	77.23/77.24		
Gesamtueberblick			147-157
Gleitkommazahlen,	80-81/80.26		
- E-formatierte	80.30		29.30
- F-formatierte	80.27		
- Vorzeichenlose F-formatierte	80.28		
GOSUB-Statement,	238.191		
- Bedingtes	239.193		
	240.194		
- Unbedingtes	238.192		
GOTO-Statement,	206-216		
	207.168		
- Bedingtes	212.170		
- Unbedingtes	207.169		
GRAPHIC INPUT-Statement			122.364
GRAPHIC PRINT-Statement			117.356
GRAPHIC PRINT USING-Statement			117
Graphikanweisung			85-134
			86.318
Grossbuchstaben	68.3a		
Grundsymbole	66-74/67.1		28
Handbuecher/Manuale			256/263-271
Hybridfunktionen,	116.68a		
- Alphanumerische	118.68c		32.68c
- Benutzerdefinierte alphanumerische	120.68i		32.68i
- Benutzerdefinierte numerische	118.68e		32.68e
- Eingebaute alphanumerische	119.68h		

	BASIC 1	BASIC 2	BASIC 3
- Eingebaute numerische	116.68d		
- Numerische	116.68b		
Identifikation			40
- ELAPSED-			40
- PROMPT-	168.117b		40
- TIMEOUT-			40
IF-Block	223.175a		19
IF EOF FL-Statement		164.294	
IF-Statement,	217.172		
- Einfaches	217.173		
Image-	187.134		
- ausgabeliste	186.130		
- kennung	187.133		66
- Statement	187.132		
- statementkennung	186.131		
Indexliste	87.42		
INF	111		
INMAX	112		
Indextrimmer		60.219	
INPUT FL-Statement		148.284	72.284
Inputliste,		149.285	
- Ergaenzte			74
Input-Output-Analyse,			197-203
- Betriebswirtschaftliche			211-213
- Volkswirtschaftliche			241-242
INPUT-Statement	167.117		
Inputueberwachung			74
Inquireangabe		162.292	81
INQUIRE BOUNDS-Statement			96.327
INQUIRE BRUSH-Statement			102.338
INQUIRE CLIP-Statement			104.339
INQUIRE FL-Statement		161.290	80.290
Inquireliste		161.291	81
INQUIRE VIEWPORT-Statement			97.328
INQUIRE WINDOW-Statement			97.329
INT (Ganzzahlbildung)	108		
INTERNAL (inhaltsstrukturiert)		120	55
Inversion		34-42/98-100	
IP	111		
I-Zeichenkette	188.137		
Kanalnummer		125.261	63
Kanalparameter		172.306	
Kanalwert		175.312	
Katalogisierungsanweisung			184.445
Kennwort	124.73	45-46	
- Diagnose-		46.205f	
- Elementar-	124.74	45	
- Erweitertes Programmsteuerungs-		46.205d	
- File-		45.205c	
- Graphik-		46.205e	
- Matrizen-		45.205b	
- Sonder-		45.205a	
- System-	124		158.403
Klammer	71.8		
Kleinbuchstabe	69.3b		
Kommentar	130.79		
Konstante,	75-83/75.17		28-29
- Alphanumerische	82.32		
- Numerische	76.18		
- Numerisch compilerexterne	76.20		
- Numerisch compilerinterne	76.19		
Konstantenliste	159.111		
Kontrollueberwachung			63
Koordinaten-			
- abfrageanweisung			95.326
- abfrageliste			97.330
- anweisung			88.319
- definitionsanweisung			88.320
- grenzliste			90.322
- grenzwert			90.323
- liste			108.344
- paar			108.345
- setzwert			97.331
Korrekturanweisung			171.420
Korrelation, Lineare	270-274		
Kostenabweichungsanalyse			214-221

	BASIC 1	BASIC 2	BASIC 3
Ladeanweisung			181.439
Laengenbegrenzung,		65	
- Obere		65.224a	
- Untere		65.224	
Laufbereichsanfang	225.179		
Laufbereichsgrenze	225.180		
Laufvariable	225.178		
LEFT$ (Linksbuendige Ausrichtung)	115		
Lehrbuecher			257-262
			271-274
LEN (Zeichenzahl im String)	117		
LET-Statement,	151.158		
- Alphanumerisches	200.160		
- Hybrid alphanumerisches	203		
- Numerisches	198.159		
let-Statement,	204.161		
- Alphanumerisches	204.163		
- Numerisches	204.162		
LINPUT-Statement	169.117c		
LINE INPUT FL-Statement			73
LINE INPUT-Statement			39
LINK-Statement		183.317	181.441
LIST-Statement			167.413
LIST PRINT-Statement			169.417
LIST PUNCH-Statement			170.418
LOAD-Statement			181.440
Loeschanweisung			174.428
LOG (Natuerlicher Logerithmus)	108		
LOG10 (Dekadischer Logerithmus)	111		
LOG2 (Binaerer Logerithmus)	111		
LOOP/EXIT-Statement	233-234		44-45
LOOP UNTIL-Statement			44
LOOP WHILE-Statement			44
Mantisse	81.31		
MARGIN-Statement			41
MARGIN FL-Statement			67
Masszahlen, Statistische	266-269		
MAT-Ausdruck,		77/ 91	
- Alphanumerischer		91.236	
- Numerischer		77.232	
MAT-Ausgabeanweisung		106.248	
MAT-Ausgabeliste,		107.251	
- DIM-begrenzte		112.256	
- DIM-gebundene		110.254	
- Numerische			112.349
MAT-Eingabeanweisung		51.209	
MAT-Eingabeliste,		52.212	
- DIM-begrenzte		59.216	
- DIM-gebundene		54.214	
- DIM-umfassende		67.226	
MAT FILL-Statement			113.350
MAT-Funktion,		77/91	
- Alphanumerische		103.245	
- Arithmetische (numerische)		94.239	
MAT-Initialisieriungsfunktion		94.240	
MAT INPUT FL-Statement		152.286	76.286
MAT-Inputliste,		68.227a	
- Erweiterte		154.287	
MAT INPUT-Statement		68.227	
MAT LET-Statement,		74.229	
- Alphanumerisches		90.234	
- Numerisches		74.230	
MAT let-Statement,		92-105	
- Alphanumerisches		103.244	
- Graphisches		105.246a	
- Numerisches		93.238	
MAT LINPUT FL-Statement			76
MAT LINPUT-Statement		72.227c	
MAT-Linputliste			
MAT OUTPUT FL-Statement		145.281	66.281
MAT OUTPUT USING FL-Statement		147.282	67.282
MAT PLOT-Statement			111.346
MAT PRINT FL-Statement			66
MAT PRINT-Statement,		107.249	
- Allgemeines		107.250	
- DIM-begrenztes		112.255	

	BASIC 1	BASIC 2	BASIC 3
- DIM-gebundenes		110.253	
MAT PRINT USING FL-Statement			67
MAT PRINT USING-Statement		114.257	
MAT READ FL-Statement			76
MAT READ-Statement,		51.210	
- Allgemeines		52.211	
- DIM-begrenztes		58.215	
- DIM-gebundenes		53.213	
- DIM-umfassendes		66,225	
MAT READ WITH FL-Statement			77-78
MAT REWRITE FL-Statement			70
MAT REWRITE WITH FL-Statement			71
Matrizen		49-50	
- addition		81/84	
- anweisung		51.208	49-53
- ausgabe	54	106-115	
- division		81/99	
- eingabe	54	51-72	
- exponentiation		82	
- Formale		172.305	
- multiplikation	56-57/231	81-86	
- quadratsumme			191
- spaltensumme			192
- rechnung			190-197
- subtraktion		81/84	
- summe			193
- zeilensumme			193
MAT-Schlussliste		108.252	
MAT-Wertzuordnungsanweisung		73.228	
MAX (Maximum zweier Zahlen)	111		
MIN (Minimum zweier Zahlen)	111		
Mittelwert	266		
MOD	111		
Modifikationsueberwachung			69
Monologbetrieb	27/58/61		160
Multisektorales Spielmodell			243-249
Name	86-87		29.41
NEXT FOR-Statement			45
NEXT LOOP-Statement			45
NEXT-Statement	225.177		
NOLIST-Statement			170.419
Normschriften			263/274-276
NR-Zeichen	188.140		
Nullmatrix		95	
null-Statement	132.82		
OPEN FL-Statement		125.260	58-60
Operator,	71.9		
- Alphanumerischer	73.14		
- Arithmetischer (numerische)	71.10		
- Boolscher	73.15		
- Dyadische (Rechenzeichen)	72.12		
- MAT-		73.233	
- Monadischer (Vorzeichen)	72.11		
- Such-			64/79
- Vergleichs-	72.13/72.13a		
ORD (Zeichenordnung)	117		
Oszillationsmodelle nach J.R. Hicks	259-265		
- Modifiziertes			249-255
OUTPUT FL-Statement		136.274	62.274
Outputliste		137.279	
Outputueberwachung			63
OUTPUT USING FL-Statement		143.280	65.280
PAGE-Statement	195.152		
PICTURE-Block			131.372
PICTURE-Statement			132.373
PLOT-Statement			107.342
Plotliste			108.343
Pointer-		123/129/136	63
- ausgangsposition		129.269	59
- position		136.276	63.276
POS (Zeichenposition im String)	117		
Preiselastizitaet der Nachfrage	254-257		
PRINT ALL-Statement			142.383
PRINT FL-Statement			62.274
PRINT-Statement,	174.123		
- Allgemeines	194.150		

	BASIC 1	BASIC 2	BASIC 3
- Graphisches			117.356
PRINT TAB-Statement	183.126		
PRINT USING FL-Statement			65.280
PRINT USING-Statement,	185.129		
- Graphisches			117.356
Prioritaetsregeln	97		
Problemanalyse	47	34-35	14-16
Problembeschreibung	47	34-35	141-16
Produkt,			
- Dyadisches		86	
- Skalares		85	
Programm-			
- ablaufplan	49		
- alternativen		43	
- aufbau	244		
- ausfuehrungsanweisung			165.410
- manipulationsanweisung			165.411
- diagnoseanweisung			138.379
- dokumentation	62		
- editierungsanweisung			163.408
- name			179.435
- planung	48-57	35-38	16-24
- segmentierungsanweisung		181.315	85
- steuerungsanweisung,	205.164		
-- Erweiterte		168.295	82-85
- test	62		25-26
- uebertragung	58		
- umsetzungsanweisung			164.409
- umwandlung	61		
Programmierung,	58-62	39-42	24-25
- Lineare			222-240
PROMPT-Identifikation	168.117b		40
Publikationen,			
- Angloamerikanische			256/257/263
- Kontinentaleuropaeische			257/262/263
OPEN FL-Statement		125.260	58-60
OPTION-Statement			37
RAD (Bogenmassberechnung)	111		
RANDOMIZE-Statement	109.59a		
READ-Statement	163.115		
READ FL-Statement			73
READ WITH FL-Statement			75-76
Rechenregeln	97-99		
Redimensionierung		95-97	50-52
REDIM-Statement			50
Regression,	270-274		202-211
- Einfache lineare	270-274		
- Multiple lineare			202-211
REM	111		
REM-Statement	131.80		
REMARK-Statement	131.81		
RENUMBER-Statement			171.421
			173.425
REPLACE-Statement			180.438
RESTORE FL-Statement		157.289	78.289
RESTORE-Statement	169.118		
RETURN-Statement	241.197		
RIGHT$ (Rechtsbuendige Ausrichtung)	115		
RND (Pseudozufallszahl)	108		
ROU (Kaufmaennisches Runden)	112		
RUN-Statement			165
REWRITE FL-Statement			69
REWRITE WITH FL-Statement			70
Satzgroesse (RECSIZE)			59
Satznummer		136.277	63
SAVE-Statement			179.434
Schrittweite	225.181		171.424
SCRATCH FL-Statement		133.272	60
SEC (Sekans)	111		
SELECT-Statement			19/46
SET BACKGROUND COLOR-Statement			100
SET BEAM-Statement			115.352
SET BOUNDS-Statement			91.321
SET BRUSH-Statement			99.334
SET CHARACTER HEIGHT-Statement			116.354
SET CLIP-Statement			101.335

	BASIC 1	BASIC 2	BASIC 3
SET COLOR-Statement			100
SET JUSTIFY-Statement			115.353
SET LINE STYLE-Statement			100
SET POSITION-Statement			114.351
SET TEXT ANGLE-Statement			116.355
SET VIEWPORT-Statement			91.324
SET WINDOW-Statement			92.325
SGN (Algebraische Vorzeichen)	108		
SIN (Sinus)	108		
SINH (Hyperbolischer Sinus)	111		
Skalarprodukt		85	
Sonderanweisung	124	48.206	49
Sonderzeichen	70-74/70.6		
Sortierprogramm	55		
Spaltenzahl	135.90	60-61	
- DIM-begrenzte		61.221	
Speicherplatzreservation	142-143		
SQR (Quadratwurzel)	108		
Sprachdarstellung,formalisierte	65-66		
Standardabweichung	267		194-195
Statement	123.70		
- Nicht numeriertes	123.72		
- Imparatives	219.175		
- inhalt	125.75		
- numierung	125-127		
- nummernliste	213.171		
- struktur	123-125		
Statistik	266-274		190-211
STOP-Statement	234.186		
String-	81-83		
- funktionsname	115.67		
- inhalt	82.34		
- klammer	82.33		
- laenge (maximale)	137.93	65	
- laengentrimmer		65.223	
STR$ (Stringumwandlung)	120		
SUB-Block		173.307	83
Subroutineanweisung		169.296	82
Subroutinename		170.298	
SUB-Statement		174.308	
Systemdiagnoseanweisung			136.376
Systemsteuerungsanweisung			161.406
TAB-Funktion	183.128		
TAN (Tangens)	108		
TANH (Hyperbolischer Tangens)	111		
Teilungszeichen	92.47c		
Teilzeichenbegrenzung	92-93		
Teilzeichenkettenbegrenzung	92.47		
TEMPLATE (feldstrukturiert)			55-56
TIME (Zeitangabe, numerisch)	111		
TIME$ (Zeitangabe, alphanumerisch)	115		
TRACE OFF-Statement			141.381
TRACE ON-Statement			138.380
Transformation-			128
- angabe			128.369
- ausdruck			128.368
- funktion			128.370
- funktionsname			128.371
Transponierung		98-99	
Trennzeichen	70.7		
Uebergabeanweisung			178.433
Unterprogrammtechnik	242-244	168-181	82-85
UNSAVE ALL-Statement			183.444
UNSAVE-Statement			183.443
USE-Statement			48
VAL (Numerischer Stringwert)	117		
Variable,	84.35		29-30
- Alphanumerische	89.44		
- Einfache alphanumerische	89.45		
- Einfache numerische	85.37/85.38		
- Indizierte	86.39/91.46		
- Indizierte alphanumerische	91.46		
- Indizierte numerische	86.39		
- Numerische	84.36		
Variablenliste,	163/197/201		
	163.116		

	BASIC 1	BASIC 2	BASIC 3
- Alphanumerische	169.117d		
	201.160b		
- Numerische	199.159b		
Variablenmaske,	187.135		
- Alphanumerische	188.143		
- Numerische	187.136		
Varianz	267		194-195
Vektorausgabe	53		
Vektoreingabe	53		
Vektor,		49-50	111-112
- Formaler		172.305	
- Variabler		68.227b	
Vereinbarung	134.85		33-39
Vergleich	218.174		
Volkswirtschaftslehre	254-265		241-255
Wertzuordnungsanweisung	196.153		
WHEN EXECEPTION-Statement			48
Wiederholungsblock			42-45
Zeichenanweisung			106.340
Zeilenzahl,	135.89		
- DIM-begrenzte		60.220	
Zielfeld,		75/90	
- Alphanumerisches		90.235	
- Numerisches		75.231	
Ziffer	69.5		
Zonenlaenge	142		
ZONEWIDTH FL-Statement			67-68
ZONEWIDTH-Statement			41
Zusatzfunktionen	110.60		
Zusatzzeichen	69.4		
Zwei-Phasen-Methode			223-240

Karl Weber/Richard Trzebiner/Horst Tempelmeier

Simulation mit GPSS

Lehr- und Handbuch zu GPSS (General Purpose Simulation System) mit wirtschaftswissenschaftlichen Anwendungsbeispielen.

489 Seiten, 72 Zeichnungen, 71 Tabellen, kart. Fr. 68.–/DM 78,–

Karl Weber/Carl Wolfram Türschmann

FOSBIC-Compiler

Ein BASIC-Compilersystem auf FORTRAN-Basis.

„BASIC-Software" 1.
224 Seiten mit vielen Zeichnungen und Compiler-Ausdrücken, kart. Fr. 40.–/DM 31,–

Karl Weber

BASIC für Anfänger

„BASIC-Software" 2.
153 Seiten mit 17 Abb. und 4 Tab., kart. Fr. 28.–/DM 31,–

Karl Weber

BASIC Programmier-Richtlinien für Austauschprogramme

„BASIC-Software" 3.
81 Seiten, kart. Fr. 30.–/DM 33,–

VERLAG PAUL HAUPT BERN UND STUTTGART

Institut für Informatik der Universität Zürich

EDV-Pflichtenhefte

Wegleitung für die Erstellung von EDV-Pflichtenheften

Computer-Beschaffungsprojekte sind anspruchsvoll und aufwendig und verlangen beträchtliche Investitionen an Geld und Arbeit. Im gesamten Beschaffungsorgan eines Computers oder einer Lösung ist das Pflichtenheft deshalb ein bedeutendes Element. Es muss ein Maximum an relevanten Informationen bieten. Aufbau und Substanz des Pflichtenheftes bestimmen sehr stark die Aussagekraft und den Wert der Offerte. Auch wird das Gelingen des geplanten Projektes nicht unwesentlich beeinflusst.

„Schriftenreihe des Instituts für Informatik der Universität Zürich" 4.
173 Seiten, kart. Fr. 25.80/DM 28,–

Institut für Informatik der Universität Zürich

EDV-Kennzahlen

Praxisbezogenes Instrumentarium zur Beurteilung der EDV-Wirtschaftlichkeit.
Mit Beiträgen von Walter J. Bruhin, Felix Brunner, Richard Heinzer, Hans Peter Koch, Gabriel G. Minder, Peter Nagel, Peter Nussbaumer, Reinhard Schretter jun., Edgar Vöhler, Heinz Waldburger.

„Lohnt sich unsere EDV eigentlich?" Um die für den Einsatz der EDV Verantwortlichen bei der Beantwortung dieser Frage zu unterstützen, hat die SVD ein Kennzahlen-System erarbeitet, das für eine betriebseigene, differenzierte und sachgerechte Beurteilung der Wirtschaftlichkeit, getrennt nach Kosten, Leistung, Nutzen, der EDV bildet.

„Schriftenreihe des Instituts für Informatik der Universität Zürich" 2. 2. Auflage
122 Seiten, kart. Fr. 24.50/DM 27,–

VERLAG PAUL HAUPT BERN UND STUTTGART

Das vorliegende Lehr- und Handbuch dient zur BASIC-Einführung und zur BASIC-Vertiefung, wobei BASIC-Standardisierungsbestrebungen (ANSI, ECMA) mitberücksichtigt werden.
In Band 3 werden im ersten Kapitel Systemanweisungen zur Verwaltung von BASIC-Programmen in benutzereigenen Programmbibliotheken und zur Editierung von BASIC-Programmen im Dialogbetrieb besprochen. Das zweite Kapitel zeigt neuere Ansätze zur BASIC-Problembearbeitung. Das dritte Kapitel befasst sich mit den Sonderanweisungen, wobei insbesondere Graphik- und Diagnoseanweisungen behandelt werden; mitberücksichtigt werden die Standardisierungsvorschläge von ANSI BASIC 79 und ANSI BASIC 82. Im vierten Kapitel werden – analog zu Band 1 – praxisbezogene BASIC-Beispiele aus dem Gebiet der Wirtschaftswissenschaften vorgestellt. Beendet wird Band 3 mit einer Buchbesprechung und einem umfangreichen Literaturverzeichnis, worin insbesondere eine ausführliche Liste der BASIC-Manuale verschiedener Hersteller von BASIC-Programmiersystemen enthalten ist.

UTB Wirtschaftswissenschaften Informatik · Datenverarbeitung

ISBN 3-258-03232-7